МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-01 Бєлка Ярослава Сергіївна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2021 р.

Науковий керівник _____ к.т.н., доц., Шендрик В.В.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«___» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Бєлка Ярослава Сергіївна
(прізвище, ім'я, по батькові)

1 Тема проекту *Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами*
затверджена наказом по університету від «___»_____2021 р. № _____

2 Термін здачі студентом закінченого проекту «___» ___грудня___ 2021 р.

3 Вхідні дані до проекту вимоги до організації процесів під час введення діяльності з підтримки прийняття рішень, зібрані від замовника.

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі та методи дослідження, концептуальне та логічне проектування сховища даних, реалізація сховища, процедур та запитів роботи з ним.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, постановка задачі, схеми OLTP та OLAP, архітектура сховища даних, схеми моделей даних зірки та сніжинки, структура гібридної мережі, діаграма потоку даних та її декомпозиція, логічні моделі бази та сховища даних, приклади результатів виконання запитів та процедур, висновки.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1.	Аналіз вимог	13.09.21-24.09.21	
2.	Планування розробки	27.09.21-01.10.21	
3.	Проектування архітектури	04.10.21-08.10.21	
4.	Концептуальне моделювання	11.10.21-22.10.21	
5.	Логічне моделювання	25.10.21-26.10.21	
6.	Створення скриптів	27.10.21-27.10.21	
7.	Написання процедур міграції	28.10.21-02.11.21	
8.	Написання процедур наповнення	03.11.21-05.11.21	
9.	Реалізація архівування	08.11.21-10.11.21	
10.	Розгортання у хмарному середовищі	11.11.21-1.11.21	
11.	Заповнення бази даних	12.11.21-16.11.21	
12.	Тестування	17.11.21-24.11.21	
13.	Оформлення документації	25.11.21-29.11.21	
14.	Прийом-передача проекту	30.11.21-30.11.21	

Магістрант _____ Бєлка Я.С.

Керівник роботи _____ к.т.н., доц. Шендрик В.В.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 102 сторінки, у тому числі 66 сторінок основного тексту, 3 сторінки списку використаних джерел та 33 сторінки додатків.

Кваліфікаційну роботу магістра присвячено розробці сховища даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами. В роботі було проведено аналіз предметної області та способів організації даних для проведення аналітики для систем підтримки прийняття рішень, сформульовано та деталізовано мету та задачі дослідження. Далі було виконано проектування сховища даних з використанням концептуального та логічного моделювання, написано скрипти для створення фізичної моделі та запити роботи з ним. Результатом проведеної роботи є сховище даних для системи підтримки прийняття рішень, яке дозволить перевірити достовірність даних і підвищити ефективність обробки і збереження даних при управлінні енергетичними мікромережами за змінних метеорологічних та географічних умов. Практичне значення роботи полягає у тому, що сховище даних стане частиною системи підтримки прийняття рішень для обробки інформації про життєвий цикл енергії при управлінні енергетичною інфраструктурою.

Ключові слова: база даних, сховище даних, системи енергоменеджменту, система прийняття рішень, відновлювана енергія, відновлювані джерела енергії.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

СД – сховище даних

OLTP (Online Transaction Processing) – онлайн обробка транзакцій.

OLAP (Online analytical processing) – онлайн аналітична обробка.

СОД – сховище операційних даних

СЕМ – системи енергоменеджменту

СППР – система прийняття рішень

ВЕ – відновлювана енергія

ВДЕ – відновлювані джерела енергії

АБ – акумуляторна батарея

СБ – сонячна батарея

ВЕУ – вітроелектроустановка

ВСТУП

Енергія життєво необхідна для стрімкого прогресу сучасного світу. Це важливий фактор, який відповідає за промисловий та сільськогосподарський розвиток. По всьому світові виробництво енергії є причиною приблизно сімдесяти відсотків викидів парникових газів. За відсутності змін у системі виробництва енергії, зміни клімату продовжать викликати незворотні зміни у навколишньому середовищі [1]. Світовий попит на енергію задовольняється використанням викопного палива, але оскільки воно наносить величезний шкідливий вплив, потрібно знайти альтернативу. Нове рішення повинно бути екологічно чистим джерелом вироблення енергії, щоб контролювати забруднення і забезпечити «зелене» середовище. Джерелами, що відповідають наведеним характеристикам є відновлювані джерела енергії.

Відновлювана енергія (ВЕ) – енергія отримувана з невичерпних, швидко поповнюваних джерел, які можуть відновлюватися природним способом. До таких джерел відносять сонячне світло, вітер, гідро-енергію (припливи та відпливи, течії). Використання цих джерел вигідне для країн та світу з екологічного та технологічного боку. Відновлювана енергія доступна для усіх країн і зокрема тих, що відчувають нестачу природної невідновлюваної сировини, такої як вугілля чи газ, які грають величезну роль на світовому ринку [2].

Актуальність: Перехід на використання зеленої енергії потребує встановлення спеціального устаткування, сонячних панелей, вітрових чи гідро-електростанцій. На їх роботу впливають різні важливі фактори, як, наприклад, швидкість вітру для вітряків та довжина світового дня, хмарність, інші погодні умови та географічне положення для сонячних панелей. Ефективне управління енергоспоживанням потребує використання системи енергоменеджменту, що складається з комплексу програмних і апаратних засобів. Процеси прогнозування та оптимізації енерговитрат супроводжуються обробкою великих масивів даних, чим більше вибірка даних, тим точніша модель

прогнозування, а отже правильніші прийняті управлінські рішення. Для збереження і обробки інформації потрібна спеціальна організація бази даних – сховище даних.

Тема: Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами.

Мета: підвищення ефективності у прийнятті рішень при управлінні енергетичними мікромережами шляхом розробки сховища даних.

Об’єкт дослідження: процес обробки та збереження даних при прийнятті рішень щодо управління життєвим циклом енергії.

Предмет дослідження: моделі та методи представлення та збереження даних у сховищі даних при управлінні енергетичною інфраструктурою.

Гіпотеза: використання розробленого сховища даних дозволить перевірити достовірність даних і підвищити ефективність обробки і збереження даних при управлінні енергетичними мікромережами за змінних метеорологічних та географічних умов.

Наукова новизна: вперше використано багатовимірну модель даних – схему «Сніжинка» для реалізації сховища даних для покращення процесу збереження та обробки інформації про життєвий цикл енергії для системи підтримки прийняття рішень (СППР) при використанні «зелених» джерел енергії.

Практичне значення: сховище даних стане частиною системи підтримки прийняття рішень для обробки інформації про життєвий цикл енергії при управлінні енергетичною інфраструктурою.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Ідентифікація проблеми

Європейський зелений курс (англ. «A European Green Deal») – взяв за мету зробити континент кліматично нейтральним до 2050 року, тож інвестиції в ініціативи підтримки політик захисту клімату можуть допомогти світу наблизитися до цієї мети – нульової позначки шкідливих викидів парникових газів в атмосферу. Саме тому, політика сприяння інвестиціям у зелену інфраструктуру, зберігання енергії та модернізації електричних мереж – це ефективний спосіб внести вклад у зниження залежності від викопного палива та глобальних викидів CO_2, а також отримати економічну вигоду [3]. Наприклад, для сприяння виробництва енергії з відновлюваних джерел в Україні передбачено три основних механізми: «зелений» тариф, пільги в оподаткуванні та пільгове приєднання до електричної мережі. «Зелений» тариф – це тариф для закупівлі електричної енергії, виробленої на об’єктах, які використовують альтернативні джерела енергії [4]. Промислові або побутові споживачі електроенергії можуть на свій розсуд встановлювати устаткування для генерації електроенергії, тим самим не тільки забезпечуючи себе потрібним обсягом енергії, але й продаючи згенерований надлишок у мережу. Для ефективного керування ресурсами потрібна система, яка б враховувала усі показники та надавала рекомендації до управління встановленими мікромережами. У цьому випадку для прийняття рішень потрібно оброблювати великі масиви інформації, саме тому необхідні дані потрібно правильно організовувати та зберігати, а для цього якнайкраще підходить сховище даних.

Перевага надається сховищу даних, а не базі даних, адже для ефективного виконання аналітики потрібне саме сховище даних, яке розробляється як шар поверх іншої бази чи баз даних. Сховище даних бере всі ці дані і створює шар, оптимізований для швидкої та простої аналітики. З визначення бази даних маємо, що це набір даних,

організований для зберігання, доступу та пошуку, а сховище даних – тип бази даних, яка об’єднує копії даних транзакцій з різноманітних вихідних систем і надає їх для аналітичного використання.

У роботі Кардона [5] (табл. 1.1-1.3) наведено порівняльну таблицю баз та сховищ даних:

Таблиця 1.1 – Порівняльна таблиця бази та сховища даних за використанням

Параметр	База даних	Сховище даних
Типи	Існують різні типи баз даних, але цей термін зазвичай застосовується до бази даних програми OLTP.	Сховище даних — це база даних OLAP, поверх OLTP або інших баз даних для виконання аналітики.
Подібність	Обидві системи OLTP і OLAP зберігають і керують даними у вигляді таблиць, стовпців, індексів, ключів, представлень і типів даних та використовують SQL для запиту даних.	
Використання	Зазвичай обмежується однією програмою: одна програма – це одна база даних. OLTP дозволяє швидко обробляти транзакції в режимі реального часу.	Зберігає дані для будь-якої кількості додатків та баз даних. OLAP дозволяє організувати одне джерело інформації для усієї організації, яке використовується для керівництва аналізом та прийняття рішень.

Операційні онлайн-системи баз даних, які виконують транзакції та обробку запитів у реальному часі, називаються системами обробки онлайн транзакцій (Online Transaction Processing (OLTP)). Наприклад, «повсякденні» операції організацій, такі як закупівля, інвентаризація, виробництво, банківська справа, реєстрація заробітної плати та бухгалтерський облік. Сховище даних служить користувачам у процесі аналізу даних та прийняття рішень. Такі системи можуть упорядковувати та представляти дані в різних

форматах відповідно до потреб різних користувачів. Ці системи відомі як системи онлайн-аналітичної обробки (Online analytical processing (OLAP)) [6]. Загалом можна припустити, що системи OLTP надають вихідні дані в сховища даних, тоді як системи OLAP допомагають їх аналізувати. На наведеному нижче діаграмі 1.1 показано продуктивність операцій OLTP та OLAP та чим вони відрізняються між бізнес-процесами та сховищем бізнес-даних [7].

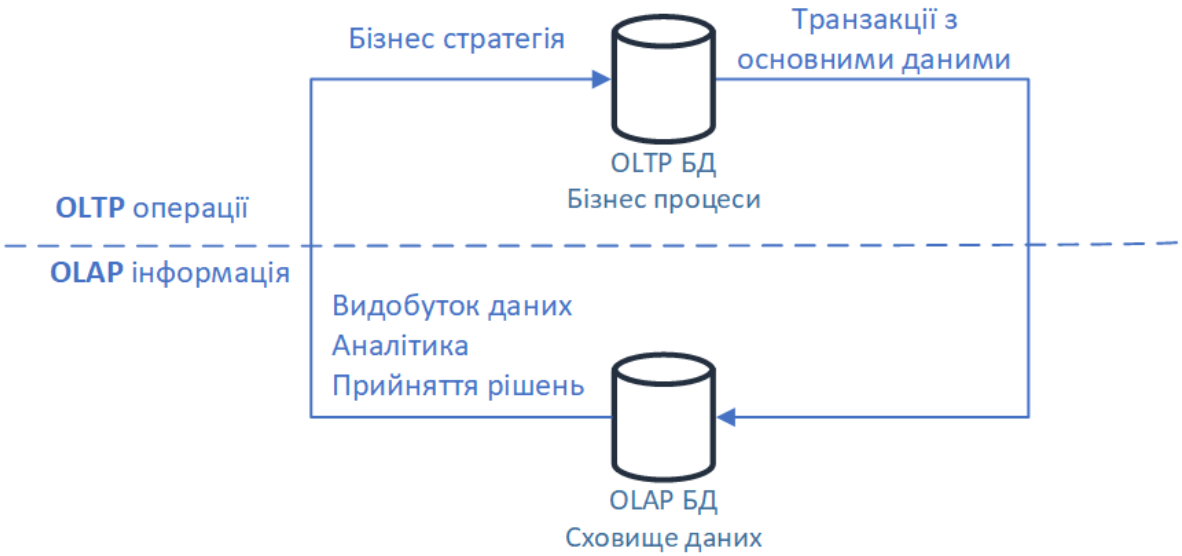


Рисунок 1.1 – OLTP та OLAP

OLTP характеризується великою кількістю коротких онлайнових транзакцій (INSERT (вставка), UPDATE (оновлення) і DELETE (видалення)). Основний акцент для систем OLTP робиться на дуже швидкій обробці запитів, підтримці цілісності даних у середовищах з доступом багатьох клієнтів та ефективності, що вимірюється кількістю транзакцій в секунду. У базі даних OLTP є детальні й поточні дані, а схема, що використовується для зберігання транзакційних баз даних – модель сутності, зазвичай, у третій нормальній формі (3NF).

OLAP характеризується відносно невеликим обсягом транзакцій. Запити часто дуже складні та передбачають агрегації. Для систем OLAP мірою ефективності є час

відгуку. У базі даних OLAP є агреговані історичні дані, що зберігаються у багатовимірних схемах.

На виконання аналітичного запиту може знадобитися кілька хвилин, що тим часом блокує доступ, тож виконання великих аналітичних запитів до OLTP бази даних є поганою практикою, адже це впливає на продуктивність системи. Оскільки база даних OLAP працює з такими великими наборами даних, вона сильно завантажує пропускну здатність центрального процесору (ЦП) і диска, саме тому використання сховища усуває навантаження на продуктивність, яке аналітика створить для транзакційної системи. У табличці 1.2 наведено узагальнення інформації про оптимізацію баз та сховищ даних для введення звітності та проведення аналізу.

Таблиця 1.2 – Порівняльна таблиця бази та сховища даних за оптимізацією

Параметр	База даних	Сховище даних
Оптимізація	Оптимізовано для виконання операцій читання-запису одноточкових транзакцій. База даних OLTP повинна забезпечувати час відповіді менше секунди.	Сховище даних призначене для обробки великих аналітичних запитів. Оптимізовано для ефективного читання/отримання великих наборів даних і для агрегування даних.
Звітність та аналіз	Через кількість об’єднань таблиць виконання аналітичних запитів дуже складне. Зазвичай вони вимагають досвіду розробника або адміністратора бази даних, знайомого з програмою.	Завдяки меншій кількості об’єднань таблиць аналітичні запити виконувати набагато легше. Напів-технічні користувачі (ті, хто вміють написати базовий SQL запит) можуть задовольнити власні потреби.

У структурі бази даних OLTP дані зберігаються у реляційній формі, в такий спосіб – це те, що забезпечує ефективність зберігання та обробки та час відповіді менше секунди, адже дані повинні бути доступні в режимі реального часу, щоб обслуговувати потреби організації тут і зараз. У структурі бази даних OLAP дані організовані спеціально для полегшення звітності та аналізу, а не для швидких транзакційних потреб. Менша кількість таблиць і простіша структура спрощують створення звітів і аналіз. Дані у сховищі оновлюються з вихідних систем за потреби (зазвичай це оновлення відбувається кожні 24 години). У табличці 1.3 наведено додаткова інформація про організацію баз та сховищ даних, а також їх відношення до угоди про рівень послуг (англ. Service-level agreement (SLA)).

Таблиця 1.3 – Порівняльна таблиця за організацією та відношенням з SLA

Параметр	База даних	Сховище даних
Організація даних	Структура бази даних OLTP містить дуже складні таблиці та об’єднання, оскільки дані нормалізовані (вона структурована таким чином, що дані не дублюються).	Дані денормалізовані, щоб збільшити час відповіді на аналітичні запити та забезпечити зручність використання для бізнес-користувачів.
Угода про рівень послуг	Бази даних OLTP зазвичай повинні забезпечувати 99,99% часу безперебійної роботи, адже база даних безпосередньо пов’язана з клієнтським додатком, збій у якому може привести до небажаних результатів.	З базами даних OLAP угоди про рівень обслуговування є більш гнучкими, оскільки очікуються випадкові простої для завантаження даних. База даних OLAP відокремлена від зовнішніх програм, що дозволяє її масштабувати.

Отже, сховище даних (СД) – це предметно-орієнтований, інтегрований, енергонезалежний набір даних, який змінюється з часом і підтримує можливість керування прийняттям рішень [8].

Сховище даних розроблене для підтримки аналізу даних і містить у собі історичні дані отримані з транзакцій. Сховище служить для аналізу історичних тенденцій і прийняття бізнес-рішень. СД відокремлює робоче навантаження аналізу від навантажень транзакцій, що допомагає підтримувати історичні записи і покращувати розуміння ділових процесів. Середовище СД включає в себе компонент витягу, перетворення та завантаження (Extract, Transform, Load (ETL)), механізм онлайн аналітичної обробки OLAP та клієнтські засоби аналізу. Технологія СД знайшла своє застосування в бізнесі, маркетингу і навіть області управління будівництвом. Зазвичай, СД включає п’ять основних компонентів [9]:

- Джерела даних, що надають вихідні дані для даних сховища, включаючи базу даних OLTP, файли даних та інші внутрішні чи зовнішні джерела даних.
- ETL процеси витягу існуючих даних з джерел, їх перетворення та завантаження в СД.
- Центральний репозиторій, що зберігає модель даних та мета-дані.
- Вітрини даних у яких зберігаються оброблені, відсортовані та реорганізовані дані.
- Доступ до даних та аналізу для осіб, які приймають рішення.

Сховища даних вилучають дані з операційних систем, вони фізично відокремлені від них, адже слугують іншій меті. Операційні системи мають бази даних і використовуються для обробки транзакцій, сховища ж даних мають окремі бази і використовуються для підтримки прийняття рішень. Дані, що зберігаються у сховищі даних володіють наступними характеристиками [10]:

- Предметна-орієнтованість – дані логічно організовані навколо основних предметів організації, наприклад, клієнтів.

- Інтегрованість – усі дані про предмет об’єднані і можуть бути проаналізовані разом.
- Змінність у часі – історичні дані зберігаються в детальній формі.
- Енергонезалежність – дані доступні лише для читання, вони не змінюються користувачами.

1.2 Аналіз методів проектування

Інтелектуальне використання електроенергії є актуальною та складною темою у дослідженні інтелектуальних мереж («smart grid»). Існуючі системи енергоменеджменту (СЕМ, з англ. EMS - Energy Management System) зазвичай надають набір автоматизованих інструментів, що використовуються операторами енергоприладів для моніторингу, контролю та оптимізації роботи системи генерації. СЕМ має за мету максимізувати продуктивність системи за рахунок наведених функцій, а для цього потрібно використовувати централізовану систему збору даних і процедури прийняття рішень [11].

У роботі Муді [12] зазначає, що сховище даних базується на метафорі «ланцюга поставок». Дані «продукт» отримують від «постачальників» даних (операційних систем або зовнішніх джерел) і тимчасово зберігають в центральному сховищі даних. Потім дані доставляються через «вітрини» даних «споживачам» (кінцевим користувачам). На рисунку 1.2 показана архітектура сховища даних (прямокутниками позначені сховища даних, а колами – процеси).

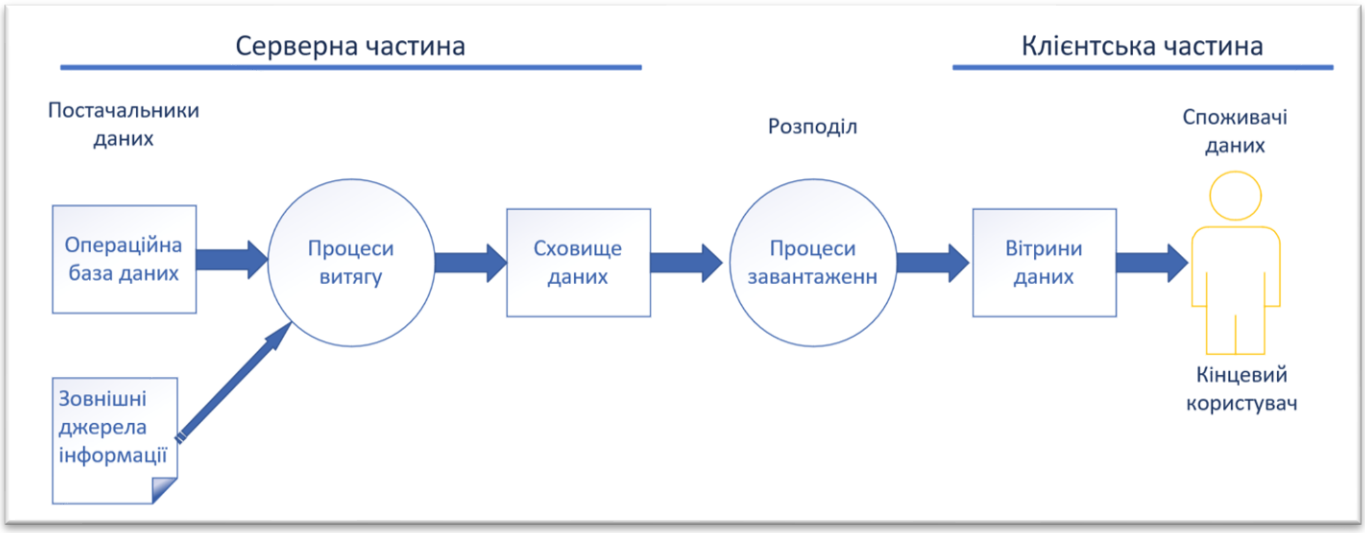


Рисунок 1.2 – Архітектура сховища даних

Архітектура складається з наступних компонентів:

- Операційна система, що реєструє деталі бізнес транзакцій. Саме тут виробляється більшість даних, необхідних для підтримки прийняття рішень.
- Зовнішні джерела інформації: наприклад, дані перепису та економічні дані для підтримки аналізу.
- Процеси витягу: ці процеси регулярно (щоденно, щотижнево) постачають інформацію до сховища даних. Дані витягуються з різних джерел, консолідуються, узгоджуються та зберігаються в єдиному форматі.
- Сховище даних працює як центральне джерело даних для підтримки прийняття рішень. Це той рівень, який використовується для забезпечення вітрин даних.
- Процеси завантаження розподіляють дані з центрального сховища по вітринам даних.
- Вітрини даних – це підмножини сховища даних, масиви інформації, яка орієнтована на конкретних користувачів. Саме з їх допомогою дані, що зберігаються у централізованому сховищі, можуть бути незалежно представлені різним користувачам. Вітрини даних представляють дані для

аналізу в зручній для користувача формі. Вони можуть бути як реальними, тими що зберігаються у вигляді реальних таблиць, заповнених з центрального сховища чи віртуальними, що визначені як представлення в центральному сховищі даних. Вітрини даних можуть бути реалізовані з використанням традиційних реляційних СУБД чи інструментів OLAP.

– Кінцеві користувачі: пишуть запити та аналізують дані, що зберігаються в вітринах даних, використовуючи інструменти запитів.

Зазвичай вирізняють два види моделей даних у сховищах даних: зірка та сніжинка. У роботі Я. Лі «Архітектура сховища даних для підтримки управління енергоспоживанням інтелектуальних електричних систем» [9] використано саме модель зірки. Ця модель має денормалізовану структуру даних, тому запити виконуються швидше. Але при цьому одна й та ж сама інформація може повторюватися у різних таблицях. На відміну від зірки, схема сніжинка має нормалізовану структуру даних, а тому займає менше дискового простору. Нормалізація таблиць приводить до зниження надмірності даних та запобіганню втратам пам’яті.

У роботі Гьогче [13] з розробки системи моніторингу та аналізу оптимізації енергоспоживання також використано схему зірка та підкреслено важливість використання сховища операційних даних. Сховище операційних даних (СОД) – база даних, призначення для інтеграції актуальних предметно-орієнтованих, змінних даних в реальному часі з різних джерел, наприклад, як мережа безпровідних датчиків чи лічильників. СОД зазвичай зберігає низкорівневі чи атомарні дані з обмеженою історією, яка фіксується в реальному чи майже реальному часі, на відміну від значно більших об’ємів даних, що зберігаються в сховищі даних. СОД – це набір змінного, деталізованого, оперативного збору даних. У дослідженні СОД тимчасово зберігає дані, зібрані мережею датчиків та лічильників. Не вся інформація про життєдіяльність енергії повинна зберігатися у сховищі даних для навчання моделей прогнозування, але в той же час змінна інформація отримана у «реальному часі» потрібна для діяльності СППР, тому СОД є підходящим рішенням для забезпечення нормальної роботи системи.

На рисунках 1.3-1.4 наведено схеми моделей даних зірка та сніжинка.

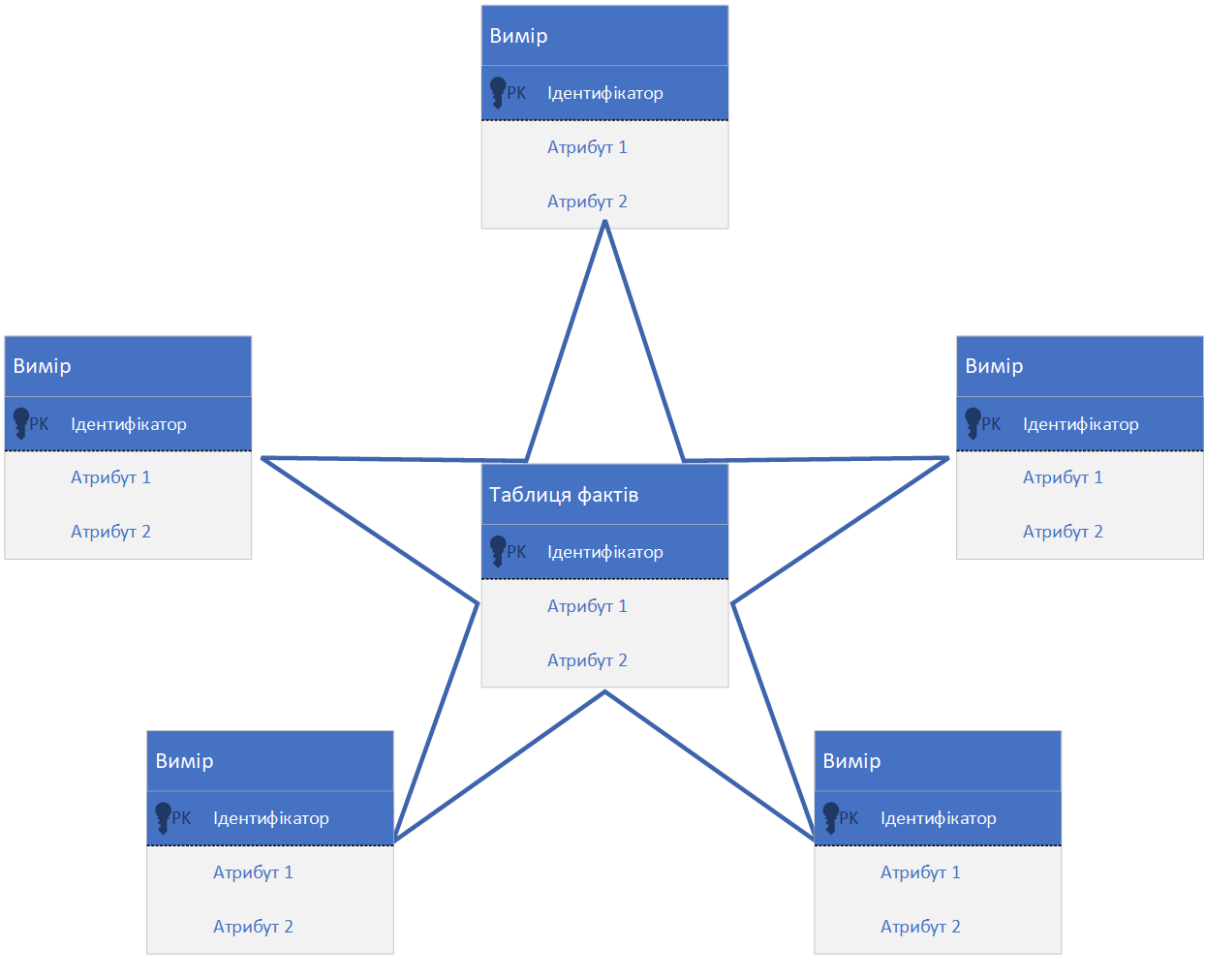


Рисунок 1.3 – Схема моделі даних – зірка

Наведена схема наочно показує чому саме таку назву отримала схема зірки. У схемі є одна центральна таблиця фактів, яку оточують таблиці вимірів. Таблиця фактів з’єднується з таблицями вимірів зовнішніми ключами. Зовнішні ключі у таблиці фактів можна використовувати, як складений первинний ключ, адже, як правило, разом зовнішні ключі однозначно визначають кожен рядок у таблиці фактів. У відносинах з вимірами таблиця фактів знаходиться на стороні "до-багатьох". У схемі зірки виміри денормалізовані, а зірка з нормалізованими вимірами стає схемою сніжинки. [14].

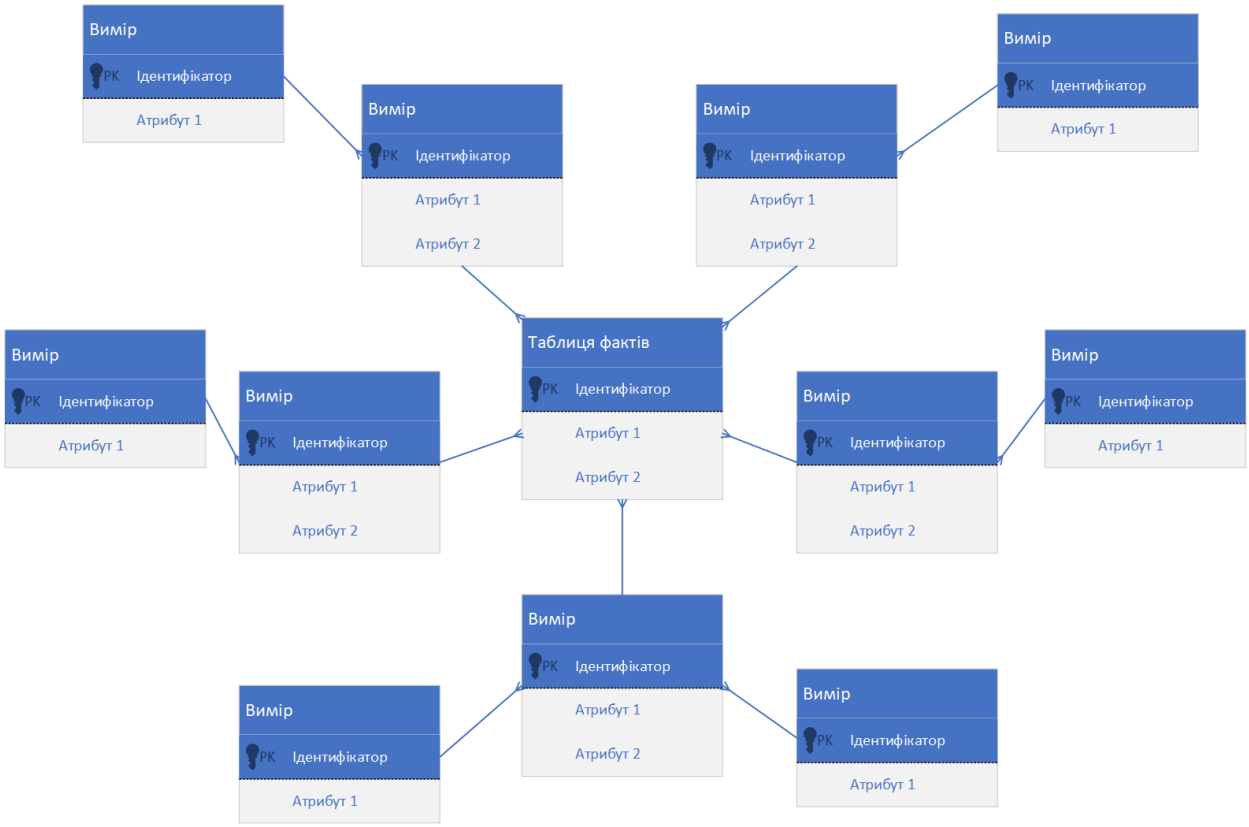


Рисунок 1.4 – Схема моделі даних – сніжинка

Схема сніжинки містить три типи таблиць: таблицю фактів, таблиці вимірів і таблиці підвимірів, які створені шляхом нормалізації таблиць вимірів, тому схему зірки можна назвати схемою сніжинки в нормалізованій формі. Завдяки цьому дані не надлишкові, без повторів, а отже потрібно менше пам’яті, що означає зменшення необхідного місця для зберігання. Існує один недолік, через збільшення кількості таблиць, запити до даних із цих таблиць будуть складнішими [15].

Щодо продуктивності схем зірки та сніжинки, з більшими таблицями фактів, що означає, наявність більшої кількісті даних та зовнішніх ключів у таблицях фактів, може знизитися продуктивність схеми зірка. Можна сказати, що чим менше зовнішніх ключів у таблиці фактів і чим більше розділені таблиці, тим швидше та ефективніше вона працюватиме. При використанні схеми зірка та сніжинка в подібному середовищі очевидно, що коли таблиці вимірів більші, краще використовувати схему сніжинки. Схема зірки містить у собі також надлишкові дані, тому обробка даних викликає

затримку в часі, розмір таблиці також більший, тому що таблиці не роз'єднані, тому опрацювання більшої таблиці є дещо ресурсозатратним процесом за часом. З іншого боку, схема сніжинки – це нормалізовані таблиці, які зберігають нормалізовані дані без надмірності, що робить виконання запитів швидшим. Тому для великих наборів даних запити у схемі зірки завжди займають більше часу на виконання, ніж у схемі сніжинки. Щодо впливу схем на розмір сховища, то нормалізація знову грає ключову роль. Сховище даних побудоване з використанням схеми зірки, яка має денормалізовані виміри, завжди буде займати більший простір, тому що вона матиме багато зайвих даних, отже, схема сніжинки, яка повністю нормалізується, потребуватиме менше місця та пам'яті, ніж схема зірки [16].

Важливим питанням у розробці сховища даних є питання про місце його розгортання. У таблиці 1.4 наведено порівняння декількох систем енергоменеджменту за їх місцем розгортання.

Таблиця 1.4 – Порівняння систем енергоменеджменту

Системи енергоменеджменту		
DEXMA Energy Intelligence	Energy Elephant	Мадек
Хмарне середовище, SaaS	Хмарне середовище, SaaS	Хостинг

Усі наведені системи Мадек, Energy Elephant, DEXMA Energy Intelligence пропонують користувачам завантажити архів історії біллінгу за електроенергію. За представленою функціональністю системи загалом націлені на зменшення витрат шляхом порівняння тарифів.

Мадек надає інструменти для створення системи біллінга. Система допоможе прийняти рішення за яким тарифом загальний платіж за електроенергію буде дешевше, розрахований за 3-х зонним тарифом (нічний, полупіковий та піковий періоди) чи за 2-х зонним (нічний та денний періоди)? Мадек порівнює споживання за різні періоди часу, порівнює платіжні програми та забезпечує облік кількості та якості електроенергії.

Energy Elephant та DEXMA Energy Intelligence схожі за функціоналом. Energy Elephant надає аудити і оцінку енергоспоживання для того, щоб зменшити витрати і споживання. На основі фактичного використання система порівнює розцінки з конкурентними цінами на ринку. Energy Elephant може показати, наприклад, що дешевше використовувати електроенергію за нічним тарифом чи газ. Також у рекомендації включені альтернативні джерела енергії, тобто порівняння вигоди проходить між сонячною панеллю чи котлом на біомасі, включно. Energy Elephant лише допоможе визначити, яке відновлюване джерело енергії може бути ефективнішим для користувача.

Рішення про розгортання на виділеному сервері чи в хмарній інфраструктурі повністю залежить від потреб організації, немає оптимального підходу для вибору. Розгортання в хмарному середовищі може дещо знизити продуктивність, однак воно забезпечує високу доступність, стійкість до відмов, практично необмежені ресурси, масштабування та еластичність ресурсів, надійність та захист, що багатократно компенсують зниження продуктивності [17].

Отже, централізоване сховище даних – це важливий ресурс для обробки інформації про життєвий цикл енергії. Збережені дані про генерацію та споживання, що записуються з устаткування, а також дані, що впливають на утворення енергії, тобто погодні чи географічні умови, можна використовувати для навчання моделі прогнозування генерації енергії. Отримані прогнози, у свою чергу, належить надавати через окремі вітрини даних, для того, щоб кожен користувач отримував лише релевантний набір рекомендацій для прийняття рішень.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі дослідження

Метою роботи є розробка сховища даних, яке буде підтримувати систему прийняття рішень при управлінні енергетичними мікромережами. Завдяки моніторингу енергоспоживання інтелектуальні служби можуть надати користувачам оптимальну конфігурацію планування ресурсів і план використання, який ґрунтується на підвищенні рівня електричних послуг [9].

Розробка проекту повинна проводитися у шість етапів, а саме:

- Аналіз вимог. На цьому етапі потрібно виконати збір вимог та аналіз даних для уточнення. Виконується детальний огляд наявної документації щодо архітектури системи та ознайомлення з існуючими базами даних. Етап завершується після перегляду результатів та затвердження.
- Планування. На етапі планування проводиться визначення цілей та завдань проекту, створення плану розробки та проводиться оцінка ризиків.
- Проектування. Етап проектування складається з проектування архітектури, концептуального моделювання предметної області та створення логічної моделі. Концептуальна модель даних повинна відображати структуру інформації, яка буде зберігатися в базі даних. Створення концептуальної моделі даних включає в себе вхідні дані зібраних вимог від замовника.
- Реалізація. Реалізація розпочинається з написання скриптів для створення операційної бази та сховища даних. Потрібно написати процедури міграції до сховища та запити для роботи з даними, а також функцію архівування. Розгортання сховище повинне проводитися у хмарному середовищі для надання можливості гнучкого масштабування за необхідністю та безпечного доступу до даних з будь-якого місця.

- Тестування. На етапі тестування заповнюється база даних та перевіряється працездатність процедур міграції, запитів та архівування.
- Завершення проекту. Заключний етап розробки завершується оформленням звітної документації та прийомом-передачею проекту.

Розробка сховища даних має бути виконана за два місяці з моменту затвердження вимог. Проект повинен бути завершеним 30 листопада 2021 року. Планування структури робіт з зазначенням календарного плану наведено у додатку А.

Сховище даних повинне бути джерелом єдиної узгодженої інформації для управління енергетичними мікромережами. Кінцеві користувачі повинні отримувати прямий доступ до даних за потребою. Користувачами системи можуть бути, як фізичні так і юридичні особи – замовники, на яких зареєстрована електрична мікромережа. Доступ до даних мікромережі можуть мати не тільки замовники, але й довірені користувачі з різними правами доступу: читач чи редактор. Також повинен бути тип користувача – адміністратор, який повинен мати доступ до керування та спостереження за потоками даних у системі. Саме тому потрібно забезпечити місце зберігання запитів користувачів. Для покращення результатів прогнозування споживання, генерації електроенергії та побудови відповідних типових моделей прогнозування потрібно розділяти типи мікромереж у залежності від локації на яких вони розташовані, таких, як наприклад:

- комунальні установи (школи, дитячі садки);
- побутові (власна будівля, дача);
- промислові (заводи зі змінним графіком роботи);
- цілодобові (заправні станції, клініки) установи.

Спроектоване сховище повинно відповідати структурі мікромереж які можуть бути зібрані при використанні відновлюваних джерел енергії (ВДЕ). ВДЕ мають економічно- та екологічно-вигідний потенціал у виробництві енергії за допомогою гібридних енергомереж (ГЕ). Поєднання різних типів ВДЕ у гібридних мережах забезпечує стабільність роботи мережі так, як типи приладів доповнюють один одного.

У роботі Шендрика С.О.[18] наведена структура гібридної мережі абонентського пункту (рис. 2.1).

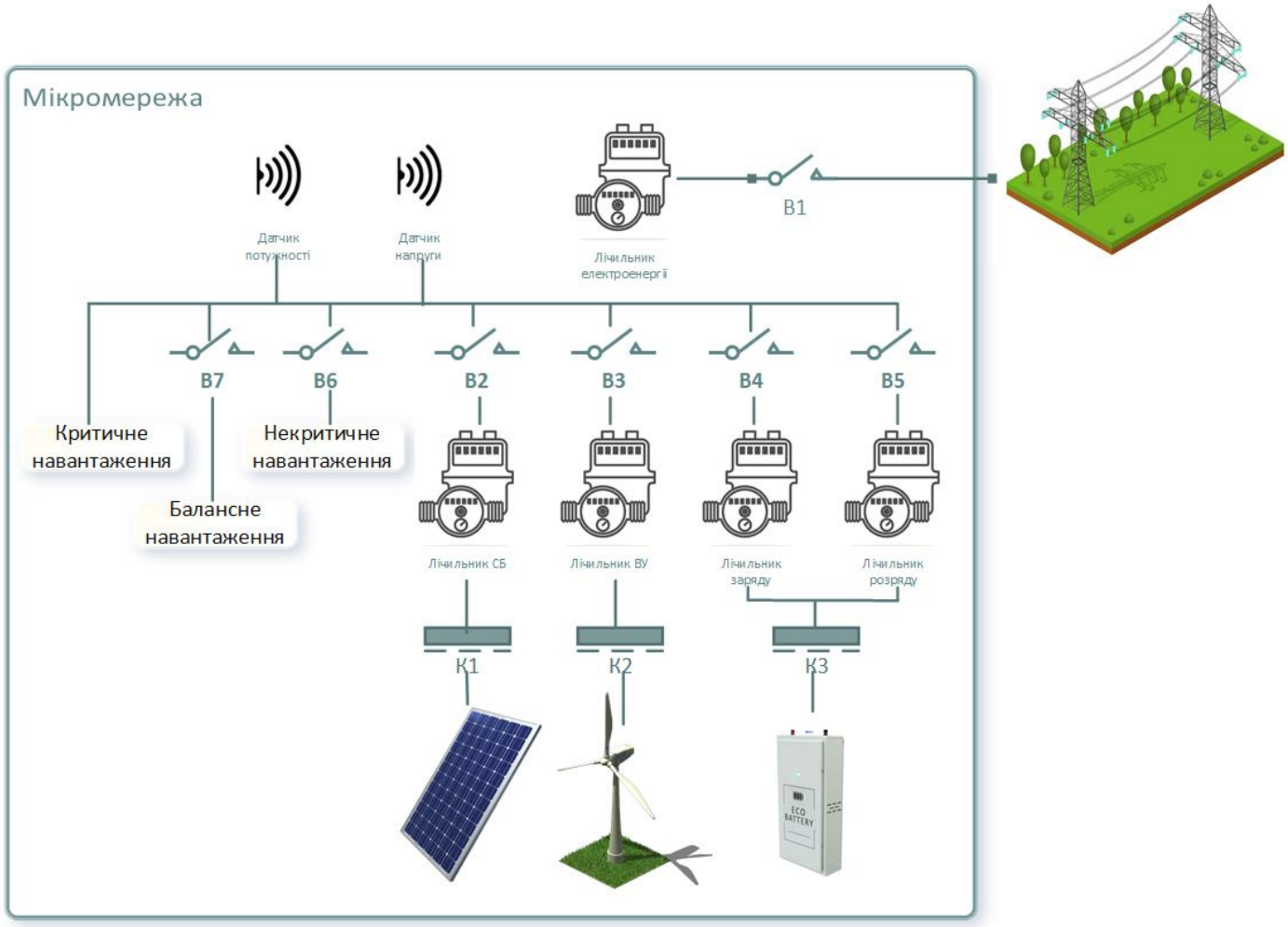


Рисунок 2.1 – Структура гібридної мережі

Тож, абонентський пункт складається з вимикача для усієї мережі – В1, лічильника електроенергії, датчиків потужності та напруги, балансного та некритичного навантаження, які під’єднані вимикачами В6 та В7 та критичного навантаження, яке не може бути відключеним. Сонячні батареї (СБ), вітроелектроустановки (ВЕУ) та акумулятори під’єднані до конвекторів К1-К3, які змінюють струм постійної напруги в змінний. У СБ та ВЕУ є свої лічильники, а у акумулятора їх два – розряду та заряду, усі вони приєднані до мережі вимикачами В2-В5.

Для того, щоб ефективно управляти ГЕ потрібно обробляти великий обсяг даних, адже прийняття рішень залежить від показників генерації, споживання електроенергії та прогнозів. Сховище даних повинне забезпечувати облік споживання та генерації (продажу) енергії згенерованої з використанням сонячних батарей чи вітряків, а також накопиченої в акумуляторах. Дані з датчиків, лічильників та конвекторів надходитимуть кожну годину, як і дані про актуальну погоду у місці розташування мікромережі. Дані про прогнозовану погоду надходитимуть з трьох годинним інтервалом. Прогнози про споживання електроенергії локацією та генерування приладами надаються на тиждень вперед. Не залежно від запитів користувачів до сховища даних повинні потрапляти лише актуальні дані та дані прогнозовані не раніше ніж за шість годин до дати та часу. Наприклад, для дати та часу «07.11.21 15:00» до сховища будуть записані лише прогнозовані дані створені «07.11.21 12:00» та «07.11.21 09:00». Це зумовлено тим, що прогнози погоди зроблені завчасно за великий проміжок часу не такі точні і не матимуть позитивного впливу на побудовані моделі прогнозування. Щотижня записи необхідні для прогнозування повинні переноситися з операційної бази до сховища даних і видалятися з попереднього вмістилища. Щорічно повинна проводитися архівація історичних даних для забезпечення швидкодії обробки поточних даних. Для реалізації усіх наведених процесів потрібно розробити процедури завантаження, передачі та очищення даних.

Отже, потрібно розробити OLTP базу даних з якою у реальному часі працювали б користувачі через систему візуалізації та OLAP сховище для зберігання історичних даних для навчання моделей прогнозування. Задачею проектування є розділення навантаження процесів аналітики та повсякденних операцій для підтримування швидкодії запитів.

Для досягнення поставленої мети потрібно вирішити відповідні завдання:

- провести аналіз предметної області;
- визначити бізнес потреби для вирішення задач користувачів;

- обрати хмарне сховище з урахуванням параметрів захисту, доступності, ефективності та ін.;
- спроектувати логічну та фізичну модель даних;
- забезпечити архівування даних;
- розробити сховище даних, процедури та запити.

2.2 Методи дослідження

Метод – це система інструкцій чи процедур, які можна виконати, щоб здійснити перехід від стартової точки (проблеми чи задачі) до цілі (рішення чи продукту). Наукові методи відрізняються від ненаукових тим, що, по-перше, наукові методи характеризуються визначеним порядком кроків (чому відповідають інструкції чи операції), по-друге, наукові методи використовуються для досягнення цілей, які прямо чи опосередковано зв’язані з нашими знаннями про світ. Вирізняють три типи наукових методів:

- Теоретичні (концептуальні).
- Емпіричні (практичні).
- Комплексні (що включають, як теоретичні, так і емпіричні методи).

Теоретичні методи загалом включають в себе інструкції, які працюють з концепціями теорії. Результатом застосування цих методів зазвичай є інше поняття (чи твердження). Теоретичні методи зосереджені на зв’язках між концептами. На противагу, емпіричні методи, окрім того що вони покладаються на певні концепції (теорії), в першу чергу пов’язані з тим, що ці концепти представляють собою в емпіричній реальності [19].

До теоретичних методів відносять такі методи:

- Аналіз. Метод аналізу застосовуються для того, щоб перейти від однієї теоретичної вихідної точки (цілого) до кінцевого стану аналізу – ідентифікації структури цілого, його всебічному вивченню.
- Класифікація. Основний принцип класифікації об’єктів полягає у виборі відповідної класифікуючої властивості, що буде використовуватися у сортуванні об’єктів по класам.
- Абстракція. Це важливий інструмент побудови та формулювання наукової теорії, теоретичної моделі, законів, обчислень і т.д. Метод абстракції дозволяє відокремитися від об’єкта з його різними властивостями та зосередитися лише на тих характеристиках, які мають відношення до мети.
- Індукція. Процес міркувань, який приводить до загального висновку, що оснований на конкретних окремих прикладах, доказах, випадків.
- Дедукція. Процес міркувань, який розпочинається з загальної істини, і застосовує цю істину до конкретного випадку.

У роботі «Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами» було використано теоретичні методи аналізу, абстракції, індукції та дедукції.

Емпіричні методи допомагають досліднику отримати та перевірити дані за допомогою сенсорного досвіду. Емпіричні методи потребують використання органів чуття і різноманітних інструментів спостереження, вимірювання та експериментування, а також засобів та методів отримання, запису, оцінки та аналізу інформації. Виділяють три фундаментальні методи емпіричних досліджень: спостереження, вимірювання та експеримент [20].

Спостереження можна охарактеризувати як навмисне, сплановане і систематичне сприйняття зовнішнього світу з використанням зору та оптичних інструментів. Спостереження є інструментом отримання емпіричної інформації про світ та перевірки емпіричних гіпотез про світ, що дозволяє систематично збирати дані, які будуть

використані для формулювання чи перевірки гіпотез. У роботі було використано емпіричний метод спостереження для формування гіпотези.

Вимірювання є методом присвоєння числових значень і одиниць вимірювання елементам емпіричної системи. В експерименті, як і у спостереженні потрібно обрати характеристику явища чи предмета, що досліджується та використовувати органи чуття та оптичні прибори чи інструменти. Однак, на відміну, в експериментах дослідник також повинен змінювати та маніпулювати умовами вихідної ситуації спостереження, контролюючи та фіксуючи зміни в результатах спостереження. Тобто, в експерименті дослідник втручається у спостережувану ситуацію, впливаючи на значення хоча б одної незалежної змінної і відстежує, чи приводить ця зміна до зміни у значенні залежної змінної (чи змінних).

3 ПРОЕКТУВАННЯ СХОВИЩА ДАНИХ

3.1 Проектування архітектури сховища даних

Базуючись на результатах аналізу було побудовано архітектуру сховища даних. Дані отримуються з «постачальників даних» (датчиків, лічильників, конвекторів пристроїв та погодних API) і зберігаються у операційній базі даних. Завдяки процесам витягу інформації, важливі накопичені дані передаються у сховище, а звідти поставляються через вітрини даних кінцевим користувачам. На рисунку 3.1 показана архітектура сховища даних.

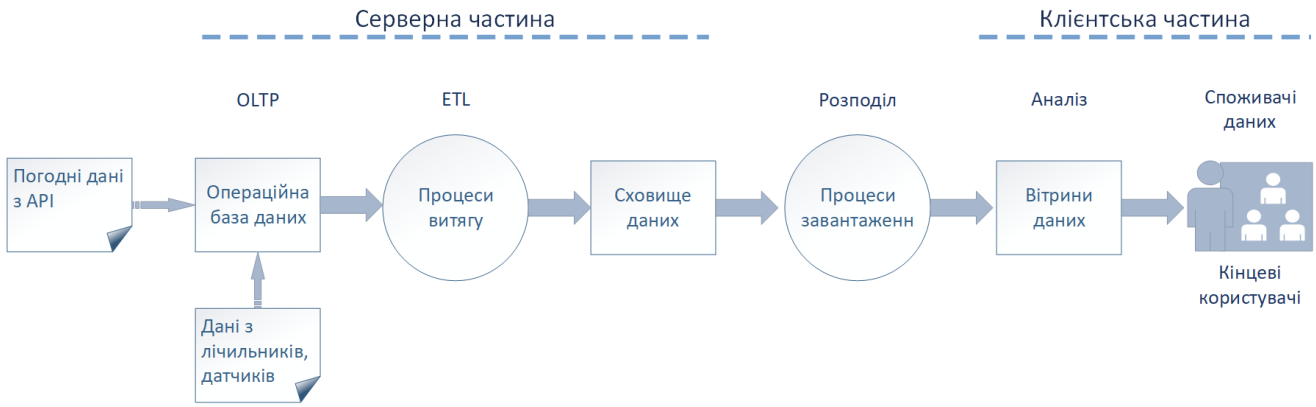


Рисунок 3.1 – Архітектура сховища даних

Архітектура складається з наступних компонентів:

– Операційна система, з якою у реальному часі працюють користувачі, до якої надходять дані з сонячних панелей, електроакумуляторів, вітроелектроустановок та погодних API, тобто усі дані, необхідні для підтримки прийняття рішень.

- Зовнішні джерела інформації: включають дані, що отримуються з зовнішніх джерел, як от дані про погоду з API, дані про споживання та генерацію з датчиків.
- Процеси витягу, які щотижнево постачають інформацію до сховища даних.
- Сховище даних.
- Процеси завантаження.
- Вітрини даних, які представляють дані для аналізу в зручній для користувача формі.
- Кінцеві користувачі.

Для представлення логічних зв’язків між межами системи, процесами та об’єктами даних на рисунку 3.2 наведено діаграму потоку даних у нотації DFD. DFD нульового рівня, або контекстна діаграма це логічна модель потоку даних через систему, яка є чудовим інструментом для узагальнення та організації детальної інформації для забезпечення логічної карти системи. Елементи діаграми потоків даних ведуть безпосередньо до фізичного проектування, де процеси відповідають програмам та процедурам, зовнішні сутності – системам з яких поступають чи куди надходять дані, а сховища даних – об’єктам, файлам та базам даних [21].

Рисунок 3.2 – Діаграма потоку даних

Для формування більш детального представлення про систему, на рисунку 3.3 наведено декомпозицію діаграми потоку даних. Розбиття узагальненого процесу контекстної діаграми на підпроцеси надає змогу виділити основні функції системи, адже

ефективність і чіткість процесів є одним з основних елементів будь-якої бізнес-операції. Діаграми потоків даних є надзвичайно корисними інструментами для підтримки цих аспектів операції.

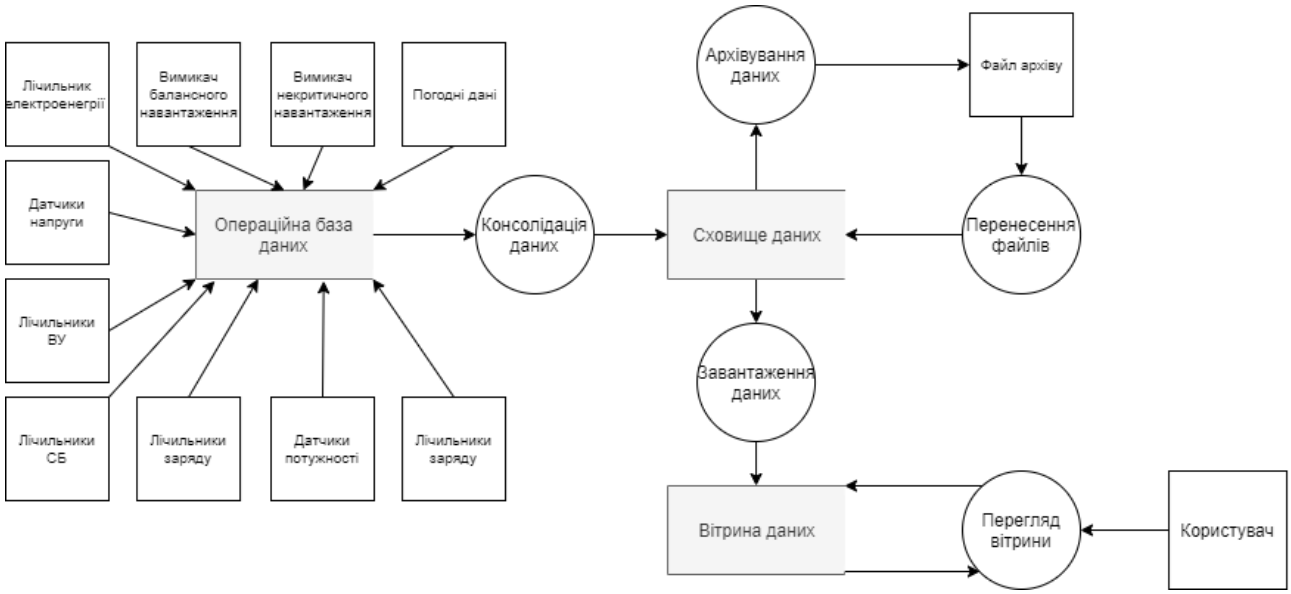


Рисунок 3.3 – Декомпозиція діаграми потоку даних

Дані до операційної бази надходять з різноманітних джерел енергомережі. З інтервалом у три години записуються прогнозовані погодні дані для надання прогнозів генерації енергії пристроями на день, три дні чи тиждень уперед. Щогодини надходять дані з приладів мікромережі. Інформація, яка потрібна для функціонування системи підтримки прийняття рішень щотижнево наповнює сховище даних. Процедура наповнення сховища повинна проходити опівночі за Грінвічем для того, щоб мінімізувати затримки у роботі усієї системи.

З метою швидкодії та збереження дискового простору дані зі сховища щорічно проходять процес архівації та записуються до .csv файлів. Правильне найменування файлів надасть змогу швидко знайти потрібний та розархівувати його за потребою. Для обслуговування діяльності певного користувача, дані постачаються до окремих вітрин даних.

3.2 Концептуальне моделювання сховища

Розробка систем зазвичай включає множину рівней абстракцій від моделювання варіантів використання до визначення класів. Моделювання баз даних традиційно включає трьорівневий підхід:

– Концептуальний рівень – документує основні сутності системи та відносин між ними.

– Логічний рівень – визначає сутності та їх відносини без деталей реалізації.

– Фізичний рівень – визначає структуру бази даних для формату специфічного для системи управління базою даних (СУБД) [22].

Під час концептуального моделювання було виділено двадцять дві сутності. Перша сутність – Customer, або Замовник. Це та людина на кого зареєстрована електрична мікромережа. Він має права редагування системи. У таблиці 3.1 представлено опис атрибутів сутності Customer.

Таблиця 3.1 – Таблиця атрибутів сутності «Customer»

Поле	Зміст
Id	Ідентифікатор замовника.
name	Ім’я.
surname	Прізвище.
login	Логін.
password	Пароль.
email	Електронна пошта.
phone	Номер телефону.

До замовника може бути додана необмежена кількість користувачів – Users, які можуть користуватися системою з доступом до даних мікромережі замовника. До цієї

сутності відносяться також адміністратори, у них відсутній замовник, натомість вони керують і переглядають дані мікромереж користувачів. У таблиці 3.2 представлено опис атрибутів сутності Users.

Таблиця 3.2 – Таблиця атрибутів сутності «Users»

Поле	Зміст
Id	Ідентифікатор користувача.
name	Ім’я.
surname	Прізвище.
login	Логін.
password	Пароль.
email	Електронна пошта.
phone	Номер телефону.
role_id	Ідентифікатор ролі користувача.
customer_id	Ідентифікатор замовника.

Користувачі можуть мати права редагування чи перегляду. У залежності від повноти чи обмежень прав у системі передбачені ролі користувачів – UserRoles. У таблиці 3.3 представлено опис атрибутів сутності UserRoles.

Таблиця 3.3 – Таблиця атрибутів сутності «UserRoles»

Поле	Зміст
Id	Ідентифікатор ролі користувача.
role_title	Назва ролі.

Для забезпечення міжнародного використання системи передбачено сутність Країни – Country. Для кожної країни зберігаються значення фазної та міжфазної напруги.

Так, наприклад, фазна напруга в Австралії та Австрії – 230 В, а в Україні, Аргентині – 220 В. У таблиці 3.4 представлено опис атрибутів сутності Country.

Таблиця 3.4 – Таблиця атрибутів сутності «Country»

Поле	Зміст
Id	Ідентифікатор країни.
name	Назва крііни.
Unm	Номінальне значення міжфазної напруги.
Unf	Номінальне значення фазної напруги.

У сутності Місто – City зберігаються координати міста (довгота та широта) для використання цих даних у запиті до API пошуку погоди, а також інформація про часовий пояс у вигляді відхилення від Грінвічу (+5 чи -5). Назва буде записуватися у форматі поштовий код, назва міста та область. Наприклад:

- «40004 Суми, Сумська область» для України;
- «19044 Horsham, PA» для США. PA – скорочення від штату Пенсильванія (англ. Pennsylvania).

У таблиці 3.5 представлено опис атрибутів сутності City.

Таблиця 3.5 – Таблиця атрибутів сутності «City»

Поле	Зміст
Id	Ідентифікатор міста.
name	Форматована назва міста.
latitude	Координати - широта.
longitude	Координати - довгота.
time_zone	Часова пояс.
country_id	Ідентифікатор країни.

Сутність Локація – Location представляє собою мікромережу, яка належить певному замовнику. У таблиці 3.6 представлено опис атрибутів сутності Location.

Таблиця 3.6 – Таблиця атрибутів сутності «Location»

Поле	Зміст
Id	Ідентифікатор локації.
latitude	Координати - широта.
longitude	Координати - довгота.
name	Назва локації.
city_id	Ідентифікатор міста.
customer_id	Ідентифікатор замовника.
location_type_id	Ідентифікатор типу локації.

Для точнішого навчання моделі виділятиметься декілька типів локацій, як, наприклад:

- Комунальний – для шкіл, дитсадків та ін.
- Побутовий – для мікромереж у домашньому використанні.
- Промисловий – для підприємств з змінним графіком роботи, як от заводи.
- Цілодобовий – для комплексів, що постійно працюють, як, наприклад, автозаправні станції (АЗС) та ін.

У таблиці 3.7 представлено опис атрибутів сутності LocationType.

Таблиця 3.7 – Таблиця атрибутів сутності «LocationType»

Поле	Зміст
Id	Ідентифікатор типу локації.
type_title	Назва типу.

У таблиці 3.8 представлено опис атрибутів сутності DateTimeInfo – дані про час та дату, що будуть використовуватися у записах про прогнозовані та актуальні дані.

Таблиця 3.8 – Таблиця атрибутів сутності «DateTimeInfo»

Поле	Зміст
Id	Ідентифікатор дати та часу.
day	День.
month	Місяць.
year	Рік.
time	Час (година:хвилини).

У сутності ActualWeather зберігатимуться дані про поточну погоду у місті. Атрибути часу сходу та заходу сонця потрібні для визначення тривалості світового дня. Дані повинні отримуватися з інтервалом у три години. У таблиці 3.9 представлено опис атрибутів сутності ActualWeather.

Таблиця 3.9 – Таблиця атрибутів сутності «ActualWeather»

Поле	Зміст
Id	Ідентифікатор погоди.
city_id	Ідентифікатор міста.
date_time_id	Ідентифікатор дати та часу.
wind_speed	Швидкість вітру.
wind_deg	Напрямок вітру.
clouds	Хмарність (у процентах).
temperature	Температура навколишнього середовища.
sunrise	Час сходу сонця у секундах.
sunset	Час заходу сонця у секундах.

Погодні API повертають напрямок вітру у градусах, поділивши коло на шістнадцять частин:

- від 348.75 до 11.25 – північний вітер;
- від 78.25 до 101.25 – східний;
- від 258.75 до 281.25 – західний;
- від 168.75 до 191.25 – південний;

Поділ на шістнадцять частин для характеристики напряму з якого дує вітер зручний для вказування чотирьох напрямів світу, а також і для допоміжних напрямів, як північно-західний чи навіть західно-північно-західний напрям.

Хмарність зберігається у процентах, що відповідає наступним станам хмарного покрову:

- Чисте небо (сонячно) – 1/8(0.125) або менше.
- Переважно сонячно – від 1/8(0.125) до 3/8(0.375).
- Хмарно з проясненнями – від 3/8(0.375) до 5/8(0.625).
- Переважно хмарно – від 5/8(0.625) до 7/8(0.875).
- Хмарно – від 7/8(0.875) до 8/8(1).

Наступна сутність ForecastWeather зберігає прогнозовану погоду та містить у собі ті ж самі атрибути, що й ActualWeather, а також атрибут для визначення часу додавання запису про прогноз. У таблиці 3.10 представлено опис атрибутів сутності ForecastWeather.

Таблиця 3.10 – Таблиця атрибутів сутності «ForecastWeather»

Поле	Зміст
Id	Ідентифікатор прогнозованої погоди.
add_date_time	Дата та час додавання запису.

У сутності DeviceType зберігатимуться три види підтримуваних пристроїв: сонячні панелі, вітроелектроустановки та акумулятори. У таблиці 3.11 представлено опис атрибутів сутності DeviceType.

Таблиця 3.11 – Таблиця атрибутів сутності «DeviceType»

Поле	Зміст
Id	Ідентифікатор типу приладу.
type_title	Назва типу приладу.

Сутність Device предназначена для зберігання масиву даних пристроїв використовуваних у мікромережах замовників. У таблиці 3.12 представлено опис атрибутів сутності Device.

Таблиця 3.12 – Таблиця атрибутів сутності «Device»

Поле	Зміст
Id	Ідентифікатор приладу.
type_id	Ідентифікатор типу приладу.
name	Назва приладу.
power	Потужність.
voltage	Напруга.

У сутності Parameters зберігається перелік параметрів до вітряків, сонячних батарей та акумуляторів. Наприклад:

- Вітряк: діаметр ротора, стартова швидкість вітру, максимальна швидкість вітру, кількість лопастей.
- Сонячна панель: мінімальна робоча температура, максимальна робоча температура.
- Акумулятор: вага, ємність.

Ідентифікатор типу приладу виключає можливість додавання до приладу невідповідного параметру. У таблиці 3.13 представлено опис атрибутів сутності Parameters.

Таблиця 3.13 – Таблиця атрибутів сутності «Parameters»

Поле	Зміст
Id	Ідентифікатор параметру.
parameter	Назва параметру.
device_type_id	Ідентифікатор типу приладу.

У сутності DeviceParameters зберігаються значення параметрів приладів. У таблиці 3.14 представлено опис атрибутів сутності DeviceParameters.

Таблиця 3.14 – Таблиця атрибутів сутності «DeviceParameters»

Поле	Зміст
Id	Ідентифікатор параметру приладу.
device_id	Ідентифікатор приладу.
parameter_id	Ідентифікатор параметру.
value	Значення параметру.

У таблиці 3.15 представлено опис атрибутів сутності LocationDevices. При додаванні нового приладу замовника, він буде записаний у сутність Device, якщо такий запис уже існує, то новий прилад у Device уже створюватися не буде, а у LocationDevices запишеться посилання на існуючий прилад. LocationDevices зберігає записи лише про сонячні панелі та вітряки.

Таблиця 3.15 – Таблиця атрибутів сутності «LocationDevices»

Поле	Зміст
Id	Ідентифікатор приладу на локації.
location_id	Ідентифікатор локації.
device_id	Ідентифікатор приладу.

Акумулятори представлені у окремій сутності, так як на відміну від попередніх типів приладів, акумулятори мають два вимикачі: розряду та заряду. У таблиці 3.16 представлено опис атрибутів сутності Accumulator.

Таблиця 3.16 – Таблиця атрибутів сутності «Accumulator»

Поле	Зміст
Id	Ідентифікатор акумулятору.
location_id	Ідентифікатор локації.
device_id	Ідентифікатор приладу.
charge	Стан лічильника заряду.
discharge	Стан лічильника розряду.
convector	Стан конвектору (ввімкнутий чи вимкнутий).

У акумулятора на відміну від сонячних панелей та вітроелектроустановок є два вимикача: заряду та розряду. У залежності від їх значення 0 – вимкнутий, 1 – увімкнений, вирізняють чотири стани акумулятора:

- Зберігання (лічильник заряду = 0, лічильник розряду = 0).
- Розрядка (лічильник заряду = 0, лічильник розряду = 1).
- Зарядка (лічильник заряду = 1, лічильник розряду = 0).
- Балансування напруги (лічильник заряду = 1, лічильник розряду = 1).

У таблиці 3.17 представлено опис атрибутів сутності LocationData. У ній зберігаються дані отримувані з вимикачів мережі, датчиків напруги та потужності. Для можливості продажу надлишкової енергії в зовнішню мережу встановлюються двонаправлені лічильники, тож вони надають показники продажу та покупки енергії у разі її нестачі.

Таблиця 3.17 – Таблиця атрибутів сутності «LocationData»

Поле	Зміст
Id	Ідентифікатор даних локації.
consumption	Споживання.
fact_sales	Продаж (віддача енергії до мережі).
purchase	Покупка енергії з мережі.
date_time_id	Ідентифікатор дати.
location_id	Ідентифікатор локації.
UA	Фазна напруга A.
UB	Фазна напруга B.
UC	Фазна напруга C.
UAB	Міжфазна напруга AB.
UAC	Міжфазна напруга AC.
UBC	Міжфазна напруга BC.
actual_power	Поточна потужність.
balance_load	Вимикач балансного навантаження.
non_critical_load	Вимикач некритичного навантаження.
electricity_meter	Вимикач лічильнику електроенергії.

У таблиці 3.18 представлено опис атрибутів сутності CurrentCapacity, яка зберігає дані про поточну ємність акумулятору.

Таблиця 3.18 – Таблиця атрибутів сутності «CurrentCapacity»

Поле	Зміст
Id	Ідентифікатор запису про акумулятор.
accumulator_id	Ідентифікатор акумулятору.
date_time_id	Ідентифікатор дати та часу.
capacity	Поточний об’єм акумулятора.

У таблиці 3.19 представлено опис атрибутів сутності ForecastConsumption з даними про прогнозоване споживання електроенергії для локації.

Таблиця 3.19 – Таблиця атрибутів сутності «ForecastConsumption»

Поле	Зміст
Id	Ідентифікатор прогнозованого споживання.
consumption	Споживання.
weather_id	Ідентифікатор погоди.
location_id	Ідентифікатор локації.

У таблиці 3.20-3.21 представлено опис атрибутів сутності ForecastData з прогнозованими та ActualData з актуальними даними генерації електроенергії для кожного приладу на локації.

Таблиця 3.20 – Таблиця атрибутів сутності «ForecastData»

Поле	Зміст
Id	Ідентифікатор прогнозованих даних генерації.
weather_id	Ідентифікатор прогнозованої погоди.
device_id	Ідентифікатор приладу.
generation	Прогноз згенерованої енергії.

Таблиця 3.21 – Таблиця атрибутів сутності «ActualData»

Поле	Зміст
Id	Ідентифікатор актуальних даних генерації.
weather_id	Ідентифікатор актуальної погоди.
device_id	Ідентифікатор приладу.
generation	Прогноз згенерованої енергії.
state	Стан приладу (ввімкнутий чи вимкнутий).
convector	Стан конвектору (ввімкнутий чи вимкнутий).

У таблиці 3.22 представлено опис атрибутів сутності Logs для збереження даних про дії (запити) користувачів.

Таблиця 3.22 – Таблиця атрибутів сутності «Logs»

Поле	Зміст
Id	Ідентифікатор запису.
query	Короткий опис запиту.
status	Статус виконання запиту.
date_time	Дата та час виконання запиту.
customer_id	Ідентифікатор користувача.

На рисунку 3.4 зображена логічна модель операційної бази даних. Модель «сутність-зв'язок» (ER-діаграма) – зручний інструмент для планування і проектування бази даних. Сутність визначає об’єкт про який зберігається інформація, а сама діаграма показує як зв’язані між собою сутності системи.

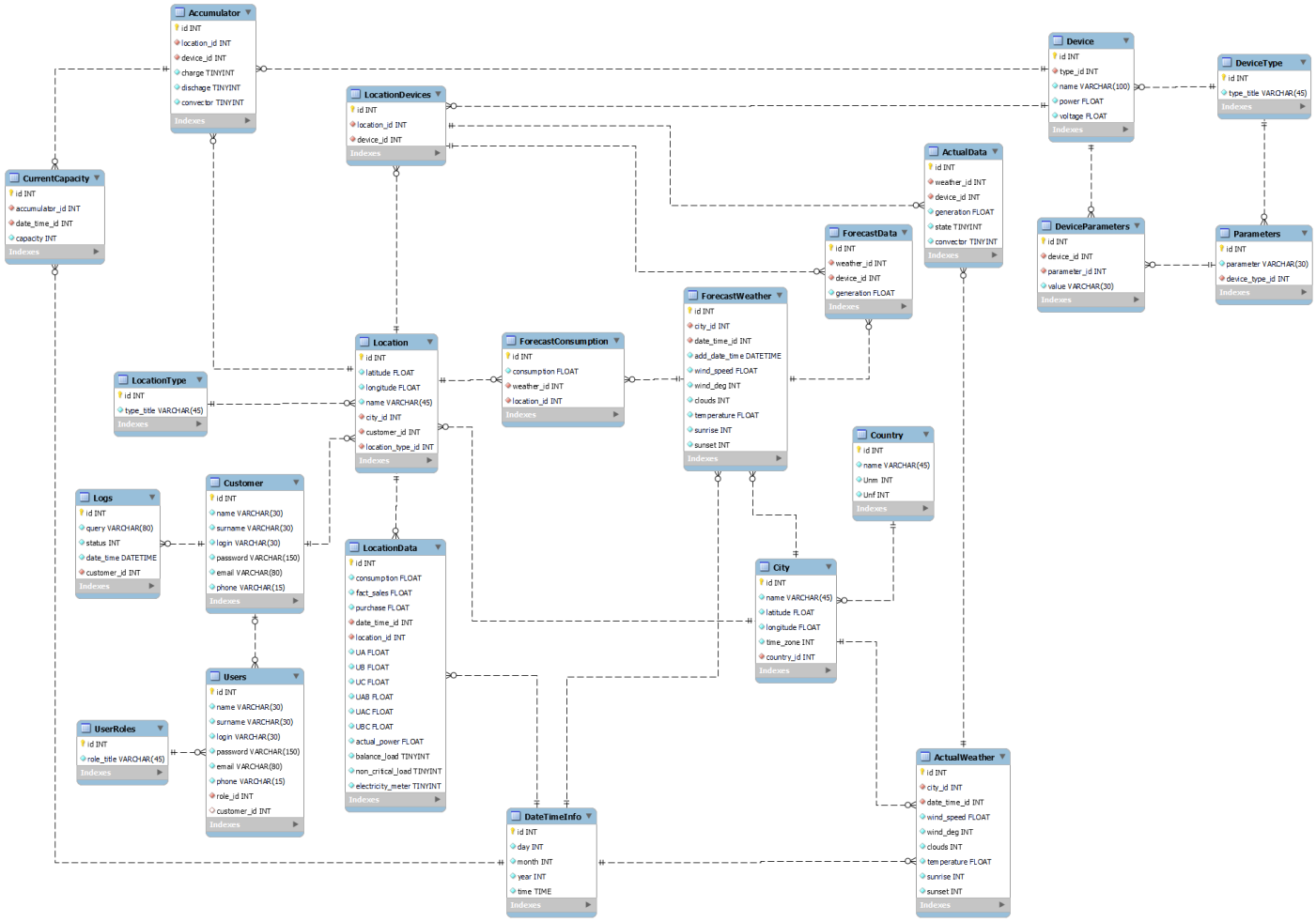


Рисунок 3.4 – Логічна модель бази даних

Якщо бази даних обробляють повсякденні транзакції, то сховища даних надають звіти і аналізи високого рівня, які дозволяють вести більш інформативну діяльність. Саме тому не вся інформація з операційної бази даних повинна зберігатися у сховищі. На рисунку 3.5 зображена логічна модель сховища даних.

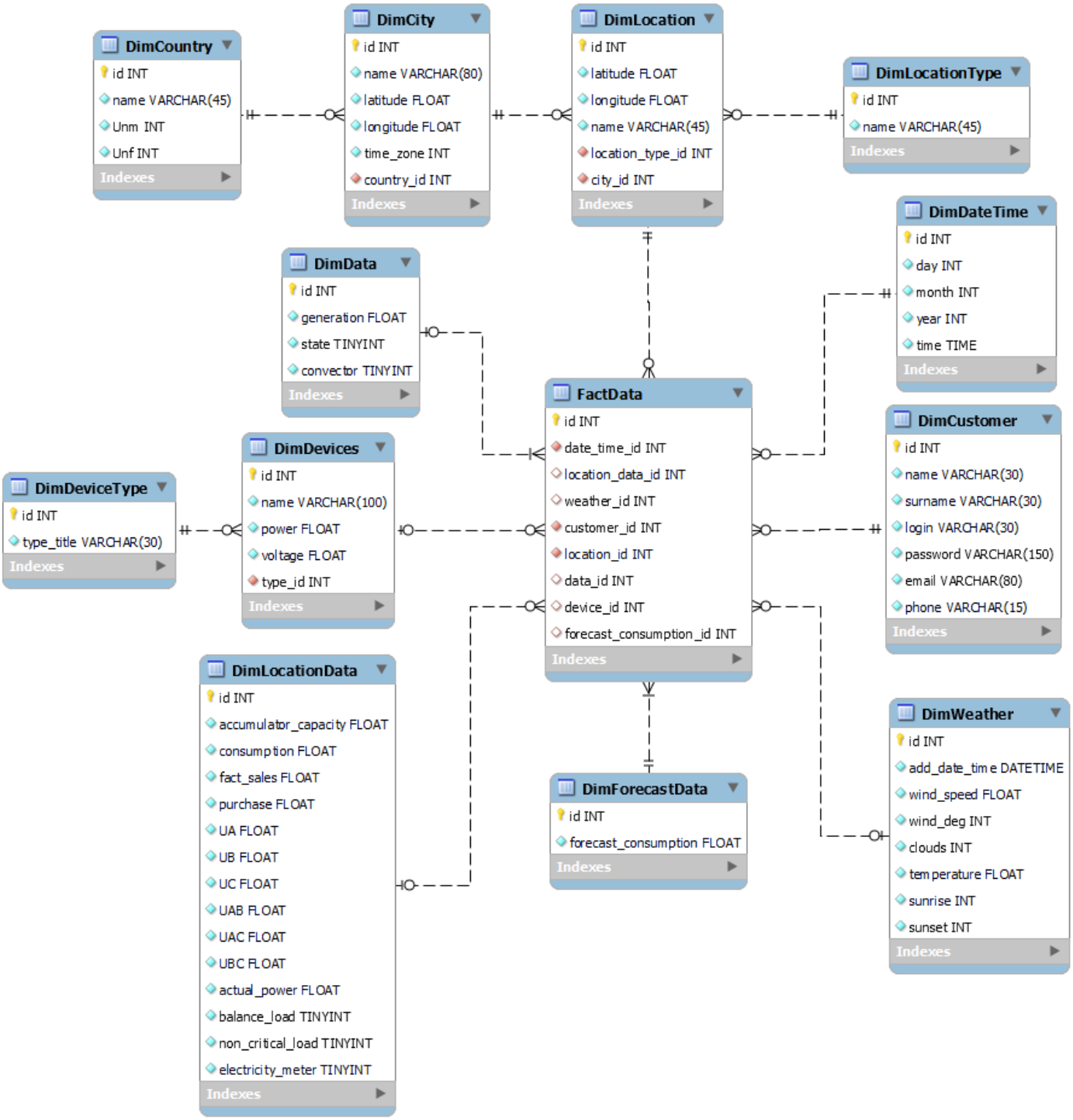


Рисунок 3.5 – Логічна модель сховища даних

Інформація у вітрини даних для кожного користувача буде формуватися з сховища даних та операційної бази, так як сховище не зберігає дані про користувачів системи, доданих до замовника.

4 РЕАЛІЗАЦІЯ СХОВИЩА ДАНИХ

4.1 Процеси завантаження, та процедури роботи зі сховищем даних

Реалізацію було розпочато з написання скриптів створення таблиць бази та сховища даних. Для реалізації та тестування було встановлено сервер MariaDB версії 10.6 (Рисунок 4.1) разом з клієнтом управління базами даних HeidiSQL. Потім, базу було наповнено тестовими даними для подальшої розробки. Лістинги коду створення та наповнення таблиць наведено у додатку Б.

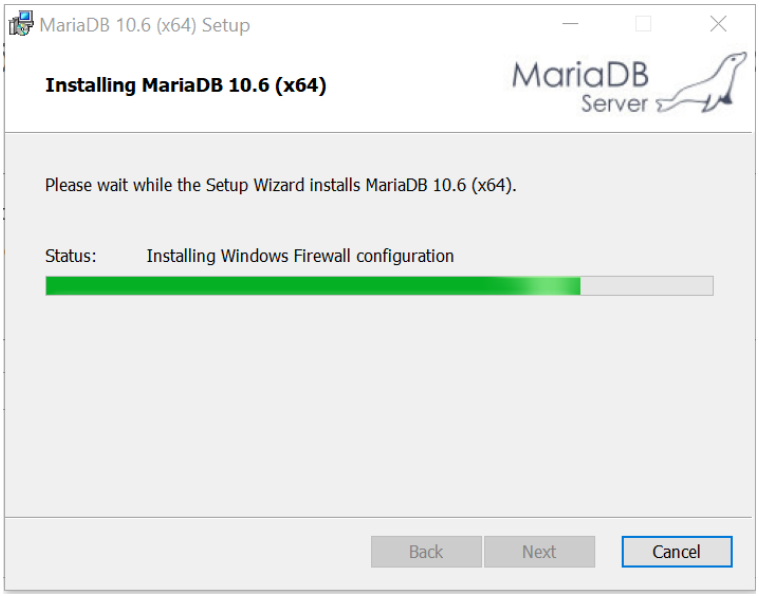


Рисунок 4.1 – Встановлення серверу MariaDB

Для наповнення сховища даних та подальшого навчання моделей прогнозування потрібні не всі дані, тож дані з таблиць Logs, Users, UserRoles, DeviceParameters, Parameters не переносяться до сховища. Хоча наповнення сховища відбувається щотижня, в той же час є дані, які повинні потрапляти до сховища після додавання до бази, що забезпечує однаковість ідентифікаторів у базі та сховищі для простішого

маніпулювання даними. З метою забезпечення цього процесу використано тригери. Тригер — це завдання, яке виконується у відповідь на певну попередньо визначену подію, яка виникла у базі даних, наприклад, після додавання нового рядка до певної таблиці. Зокрема, ця подія передбачає вставку, зміну або видалення даних таблиці, і тригер може спрацювати до або відразу після будь-якої такої події [23].

На рисунках 4.2-4.3 зображено тригери для додавання даних з таблиць бази Customer та DateTimeInfo до таблиць сховища DimCustomer та DimDateTime відповідно.

Рисунок 4.2 – Тригери для таблиці замовників

Рисунок 4.3 – Тригери для таблиці дати та часу

На рисунку 4.4 зображено тригер для додавання даних з таблиць бази Device та DeviceType до таблиць сховища DimDevices та DimDeviceType. Завдячуючи структурі сховища даних, інформація щодо того до якої локації належить прилад буде об’єднуватися з власне приладом у таблиці FactData, тому таблиця бази LocationDevices не переноситься до сховища.

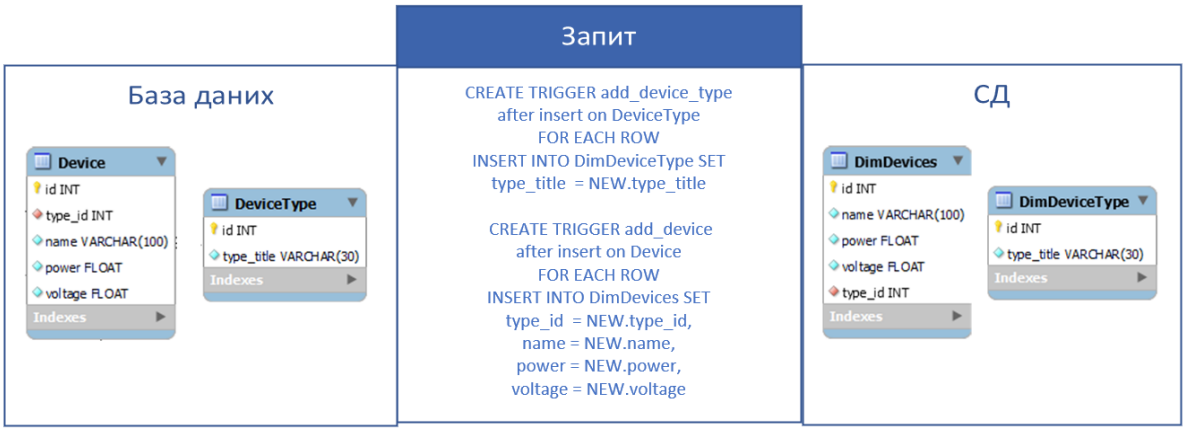


Рисунок 4.4 – Тригери для таблиць приладів та типів приладів

Ще чотири таблиці бази переходять до сховища без змін. Це таблиці City та Country, дані з яких вставляються до таблиць DimCity та DimCountry у сховищі (Рисунок 4.5), а також таблиці бази Location, LocationType і відповідні їм у сховищі таблиці DimLocation та DimLocationType.

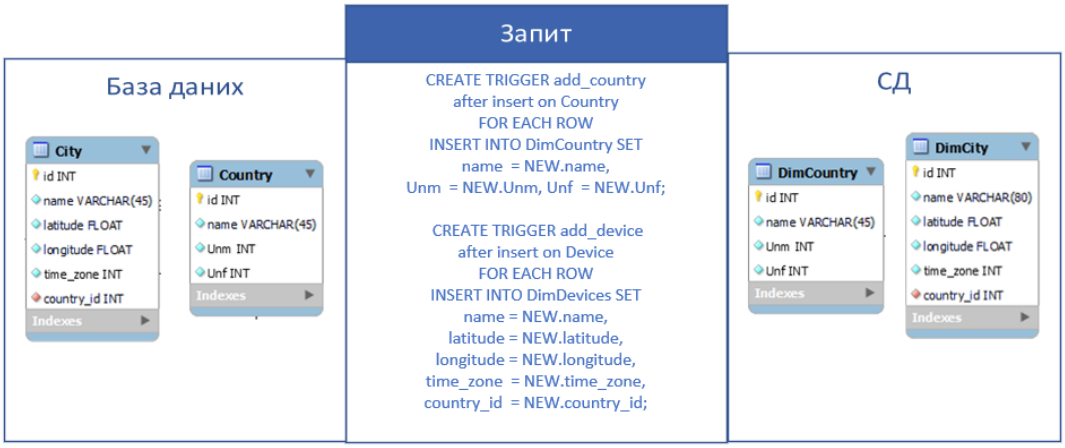


Рисунок 4.5 – Тригери для таблиць країн та міст

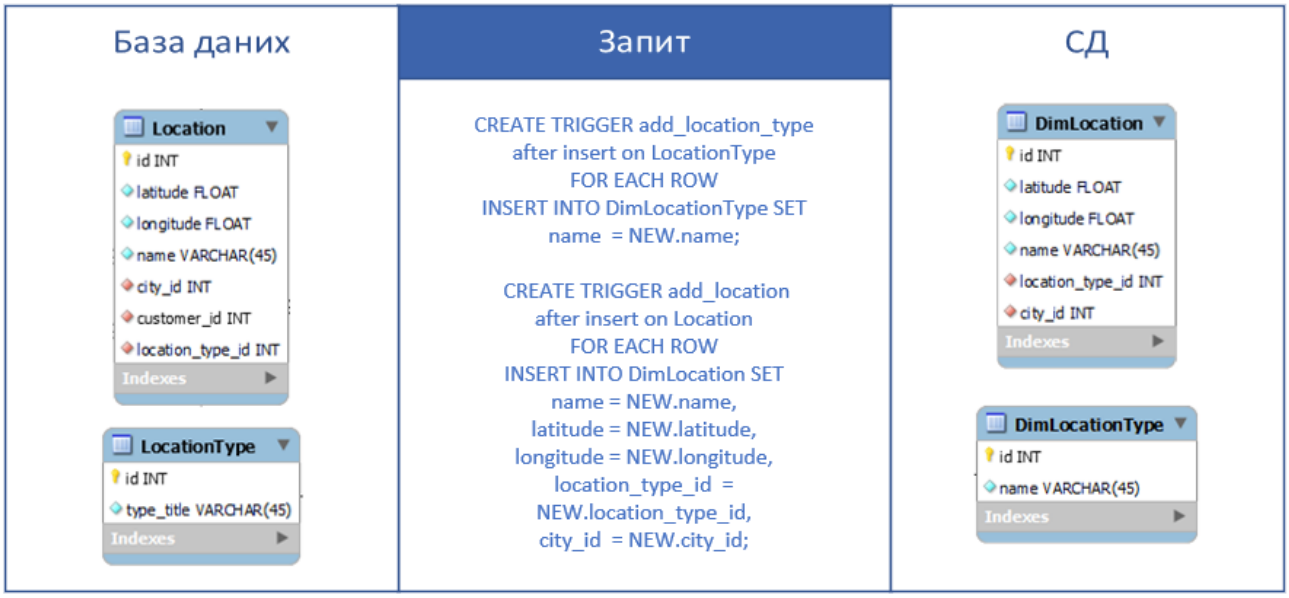


Рисунок 4.6 – Тригери для таблиці локацій та типів локацій

Тригер створюється ключовими словами «Create Trigger» після яких слідує назва тригеру. Наступним параметром є час виконання тригеру («before» чи «after»), подія («insert» – вставка, «update» – оновлення чи «delete» – видалення) та таблиця у якій подія виникла. Останнім є опис дії, що потрібно виконати. Лістингу коду тригерів наведені у додатку В.

Відповідність даних, що залишилися у базі та вимірів сховища показана у табличці 4.1. Для цих даних, що повинні передаватися до сховища щотижня передбачена процедура після виконання якої будуть заповнюватися не тільки таблиці вимірів, але й головна таблиця фактів у якій поєднані всі записи.

Таблиця 4.1 – Відповідності даних бази та сховища

База даних	Сховище даних
ActualWeather, ForecastWeather	DimWeather
Accumulator, CurrentCapacity, LocationData	DimLocatioData
ForecastConsumption	DimForecastData
ForecastData, ActualData	DimData

Таблиця фактів складається з ідентифікатору та восьми атрибутів, з яких три date_time_id, customer_id, location_id мають обмеження «Not Null» – не пустий. Значення атрибутів таблиці фактів визначають тип запису. Серед атрибутів вирізняються чотири date_time_id, customer_id, location_id, device_id саме тим, що ідентифікатори цих записів відповідають ідентифікаторам бази даних. Інші ж ідентифікатори створюються при виконанні процедури перенесення даних.

Таблиця 4.2 – Типи записів у таблиці фактів FactTable

Атрибут		Тип запису		
	Тип 1	Тип 2.1	Тип 2.2	Тип 3
date_time_id	Not null	Not null	Not null	Not null
location_data_id	Not null	Null	Null	Null
weather_id	Null	Not null	Not null	Not null
customer_id	Not null	Not null	Not null	Not null
location_id	Not null	Not null	Not null	Not null
data_id	Null	Not null	Not null	Null
device_id	Not null	Not null	Not null	Null
forecast_consumption id	Null	Null	Null	Not null

У таблиці відображено наступні типи записів:

- Тип 1 – запис про актуальні дані з датчиків локації.
- Тип 2.1 – запис про актуальну генерацію енергії для кожного приладу на локації.
- Тип 2.2 – прогнозовану генерацію енергії для кожного приладу на локації.
- Тип 3 – запис про прогнозоване споживання електроенергії на локації.

Структура процедури для наповнення сховища даних (Рисунок 4.7) складається з трьох функцій: FunctionAddActualLocationData – додає записи типу 1, FunctionAddActualDeviceData – записи типу 2.1, FunctionAddForecastData – працює з

записами типів 2.2 та 3, бо прогнозовані дані об’єднані записами про прогнозовану погоду. До процедури та функцій передаються два параметри start_ та end_, які визначають проміжок часу, записи за який повинні додатися до сховища. Лістинги усіх подальше представлених процедур, функцій, тощо наведені у додатку В.

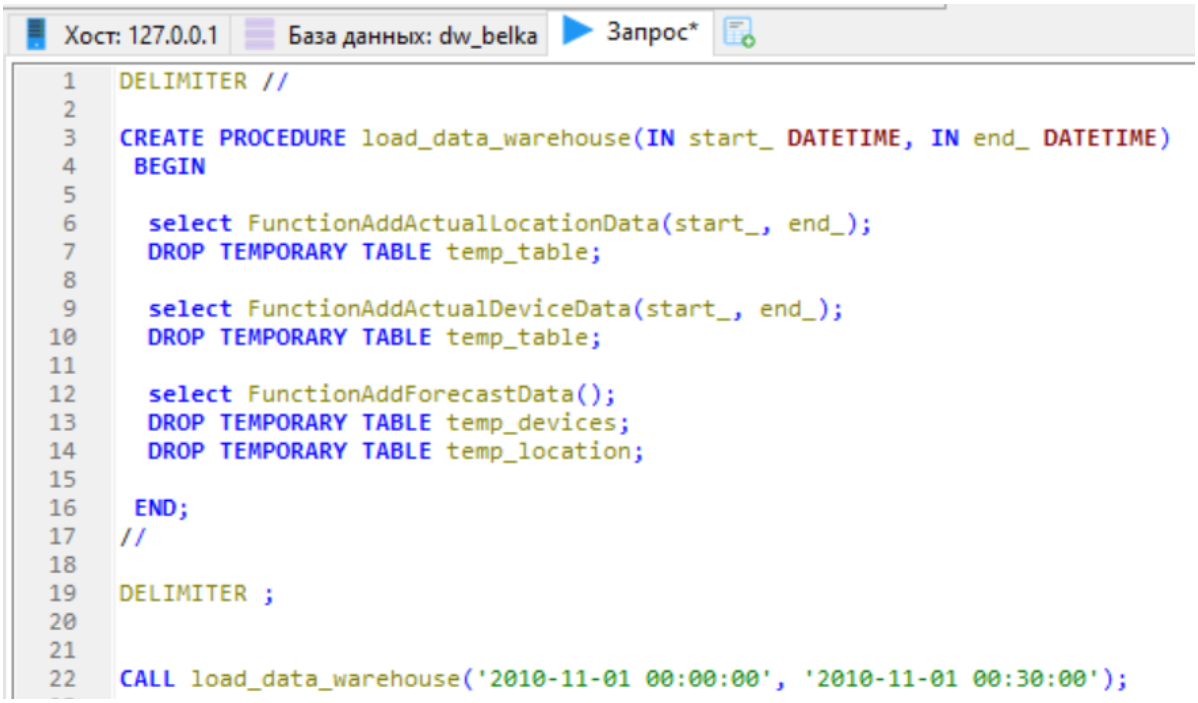
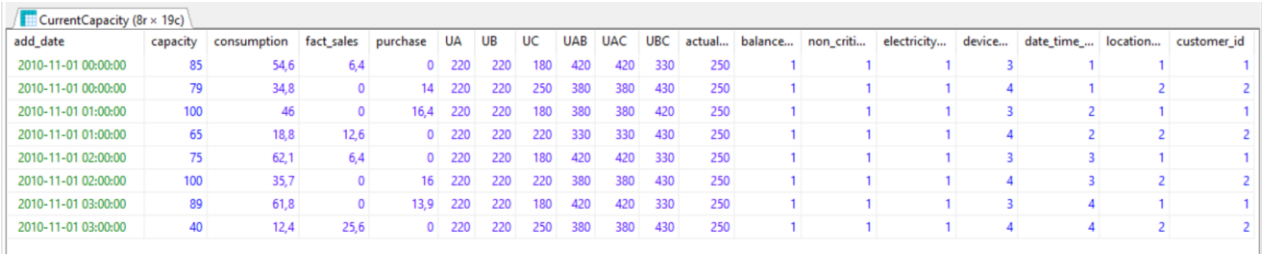


Рисунок 4.7 – Процедура наповнення сховища даних

У кожній функції створюються тимчасові таблиці «Create temporary table». У цьому випадку створення тимчасових таблиць – тривалість життя яких дорівнює лише поточному сеансу дуже корисне. Вони знадобляться при виконанні множини запитів до особливо великої таблиці. Замість того, щоб повторно виконувати усі запити до великої таблиці, можна створити тимчасову і потім виконувати запити до неї. Також, тимчасові таблиці, використані у функціях, характеризуються тим, що вони створені з використанням об’єднань «Join», тому робота з тимчасовою таблицею зручніша. У першій функції FunctionAddActualLocationData для додавання до сховища записів про актуальні дані з датчиків локації створюється тимчасова таблиця «temp_table» (Рисунок 4.8).

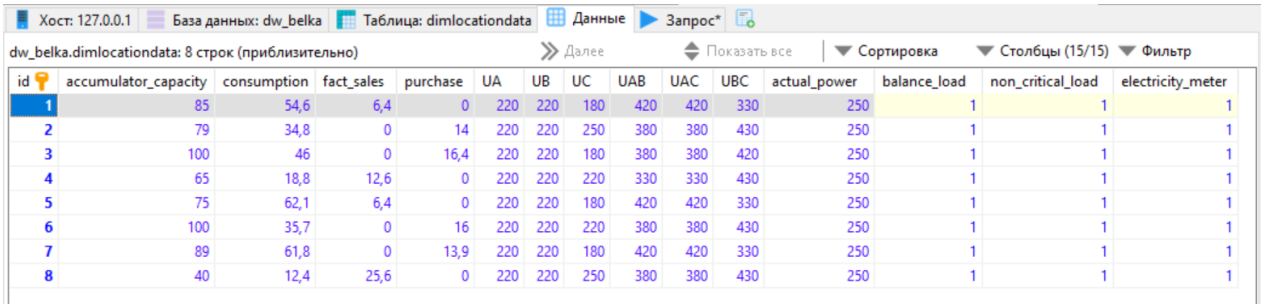


CurrentCapacity (8r × 19c)

add_date	capacity	consumption	fact_sales	purchase	UA	UB	UC	UAB	UAC	UBC	actual...	balance...	non_criti...	electricity...	device...	date_time_...	location...	customer_id
2010-11-01 00:00:00	85	54,6	6,4	0	220	220	180	420	420	330	250	1	1	1	3	1	1	1
2010-11-01 00:00:00	79	34,8	0	14	220	220	250	380	380	430	250	1	1	1	4	1	2	2
2010-11-01 01:00:00	100	46	0	16,4	220	220	180	380	380	420	250	1	1	1	3	2	1	1
2010-11-01 01:00:00	65	18,8	12,6	0	220	220	220	330	330	430	250	1	1	1	4	2	2	2
2010-11-01 02:00:00	75	62,1	6,4	0	220	220	180	420	420	330	250	1	1	1	3	3	1	1
2010-11-01 02:00:00	100	35,7	0	16	220	220	220	380	380	430	250	1	1	1	4	3	2	2
2010-11-01 03:00:00	89	61,8	0	13,9	220	220	180	420	420	330	250	1	1	1	3	4	1	1
2010-11-01 03:00:00	40	12,4	25,6	0	220	220	250	380	380	430	250	1	1	1	4	4	2	2

Рисунок 4.8 – Тимчасова таблиця у функції FunctionAddActualLocationData

На рисунках 4.9-4.10 проілюстровано роботу першої функції та формат даних, які записуються до виміру DimLocationData та таблиці фактів FactData.

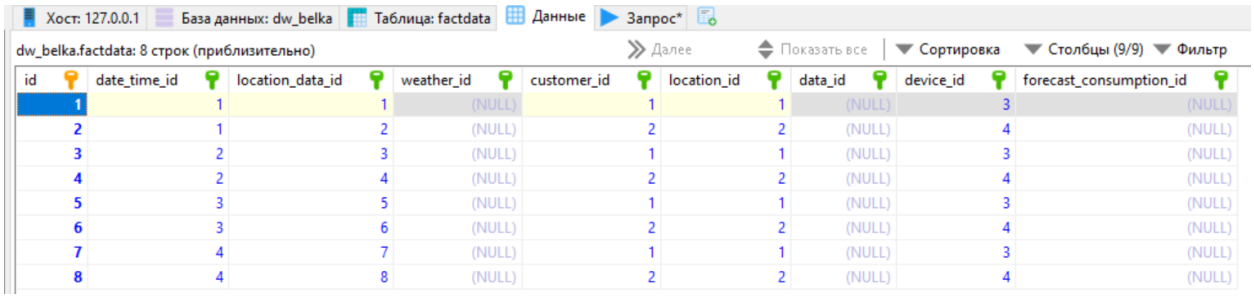


Хост: 127.0.0.1 | База данных: dw_belka | Таблица: dimlocationdata | Данные | Запрос*

dw_belka.dimlocationdata: 8 строк (приблизительно) — Далее | Показать все | Сортировка | Столбцы (15/15) | Фильтр

id	accumulator_capacity	consumption	fact_sales	purchase	UA	UB	UC	UAB	UAC	UBC	actual_power	balance_load	non_critical_load	electricity_meter
1	85	54,6	6,4	0	220	220	180	420	420	330	250	1	1	1
2	79	34,8	0	14	220	220	250	380	380	430	250	1	1	1
3	100	46	0	16,4	220	220	180	380	380	420	250	1	1	1
4	65	18,8	12,6	0	220	220	220	330	330	430	250	1	1	1
5	75	62,1	6,4	0	220	220	180	420	420	330	250	1	1	1
6	100	35,7	0	16	220	220	220	380	380	430	250	1	1	1
7	89	61,8	0	13,9	220	220	180	420	420	330	250	1	1	1
8	40	12,4	25,6	0	220	220	250	380	380	430	250	1	1	1

Рисунок 4.9 – Вимір DimLocationData



Хост: 127.0.0.1 | База данных: dw_belka | Таблица: factdata | Данные | Запрос*

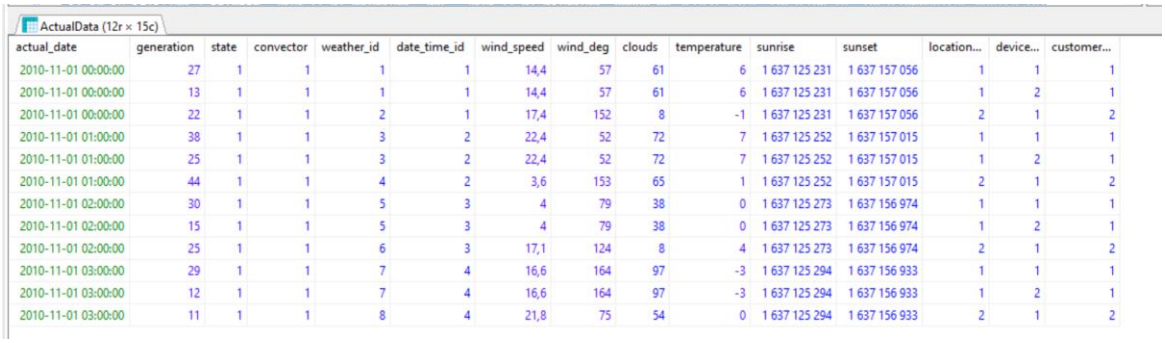
dw_belka.factdata: 8 строк (приблизительно) — Далее | Показать все | Сортировка | Столбцы (9/9) | Фильтр

id	date_time_id	location_data_id	weather_id	customer_id	location_id	data_id	device_id	forecast_consumption_id
1	1	1	(NULL)	1	1	(NULL)	3	(NULL)
2	1	2	(NULL)	2	2	(NULL)	4	(NULL)
3	2	3	(NULL)	1	1	(NULL)	3	(NULL)
4	2	4	(NULL)	2	2	(NULL)	4	(NULL)
5	3	5	(NULL)	1	1	(NULL)	3	(NULL)
6	3	6	(NULL)	2	2	(NULL)	4	(NULL)
7	4	7	(NULL)	1	1	(NULL)	3	(NULL)
8	4	8	(NULL)	2	2	(NULL)	4	(NULL)

Рисунок 4.10 – Таблиця фактів FactData

Таблиця фактів повністю відповідає своїй назві, бо об’єднує через зовнішні ключі усі дані тим самим однозначно ідентифікуючи запис та знання про нього.

У другій функції FunctionAddActualDeviceData для додавання до сховища записів про актуальну генерацію енергії для кожного приладу на локації створюється тимчасова таблиця «temp_table» (Рисунок 4.11).

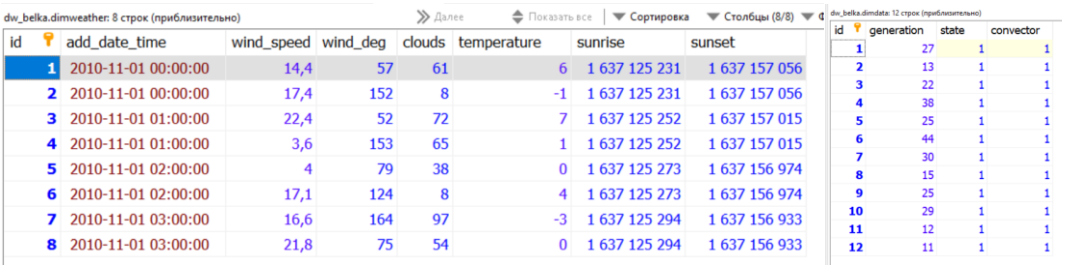


ActualData (12r × 15c)

actual_date	generation	state	convector	weather_id	date_time_id	wind_speed	wind_deg	clouds	temperature	sunrise	sunset	location...	device...	customer...
2010-11-01 00:00:00	27	1	1	1	1	14,4	57	61	6	1 637 125 231	1 637 157 056	1	1	1
2010-11-01 00:00:00	13	1	1	1	1	14,4	57	61	6	1 637 125 231	1 637 157 056	1	2	1
2010-11-01 00:00:00	22	1	1	2	1	17,4	152	8	-1	1 637 125 231	1 637 157 056	2	1	2
2010-11-01 01:00:00	38	1	1	3	2	22,4	52	72	7	1 637 125 252	1 637 157 015	1	1	1
2010-11-01 01:00:00	25	1	1	3	2	22,4	52	72	7	1 637 125 252	1 637 157 015	1	2	1
2010-11-01 01:00:00	44	1	1	4	2	3,6	153	65	1	1 637 125 252	1 637 157 015	2	1	2
2010-11-01 02:00:00	30	1	1	5	3	4	79	38	0	1 637 125 273	1 637 156 974	1	1	1
2010-11-01 02:00:00	15	1	1	5	3	4	79	38	0	1 637 125 273	1 637 156 974	1	2	1
2010-11-01 02:00:00	25	1	1	6	3	17,1	124	8	4	1 637 125 273	1 637 156 974	2	1	2
2010-11-01 03:00:00	29	1	1	7	4	16,6	164	97	-3	1 637 125 294	1 637 156 933	1	1	1
2010-11-01 03:00:00	12	1	1	7	4	16,6	164	97	-3	1 637 125 294	1 637 156 933	1	2	1
2010-11-01 03:00:00	11	1	1	8	4	21,8	75	54	0	1 637 125 294	1 637 156 933	2	1	2

Рисунок 4.11 – Тимчасова таблиця у функції FunctionAddActualDeviceData

На рисунках 4.12-4.13 проілюстровано роботу другої функції та формат даних, які записуються до вимірів DimWeather та DimData та таблиці фактів FactData.



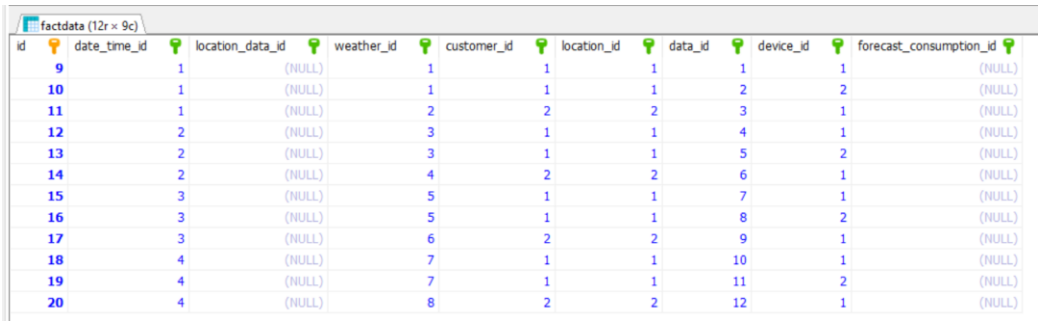
dw_belka.dimweather: 8 строк (приблизительно)

id	add_date_time	wind_speed	wind_deg	clouds	temperature	sunrise	sunset
1	2010-11-01 00:00:00	14,4	57	61	6	1 637 125 231	1 637 157 056
2	2010-11-01 00:00:00	17,4	152	8	-1	1 637 125 231	1 637 157 056
3	2010-11-01 01:00:00	22,4	52	72	7	1 637 125 252	1 637 157 015
4	2010-11-01 01:00:00	3,6	153	65	1	1 637 125 252	1 637 157 015
5	2010-11-01 02:00:00	4	79	38	0	1 637 125 273	1 637 156 974
6	2010-11-01 02:00:00	17,1	124	8	4	1 637 125 273	1 637 156 974
7	2010-11-01 03:00:00	16,6	164	97	-3	1 637 125 294	1 637 156 933
8	2010-11-01 03:00:00	21,8	75	54	0	1 637 125 294	1 637 156 933

dw_belka.dimdata: 12 строк (приблизительно)

id	generation	state	convector
1	27	1	1
2	13	1	1
3	22	1	1
4	38	1	1
5	25	1	1
6	44	1	1
7	30	1	1
8	15	1	1
9	25	1	1
10	29	1	1
11	12	1	1
12	11	1	1

Рисунок 4.12 – Виміри DimWeather та DimData



factdata (12r × 9c)

id	date_time_id	location_data_id	weather_id	customer_id	location_id	data_id	device_id	forecast_consumption_id
9	1	(NULL)	1	1	1	1	1	(NULL)
10	1	(NULL)	1	1	1	2	2	(NULL)
11	1	(NULL)	2	2	2	3	1	(NULL)
12	2	(NULL)	3	1	1	4	1	(NULL)
13	2	(NULL)	3	1	1	5	2	(NULL)
14	2	(NULL)	4	2	2	6	1	(NULL)
15	3	(NULL)	5	1	1	7	1	(NULL)
16	3	(NULL)	5	1	1	8	2	(NULL)
17	3	(NULL)	6	2	2	9	1	(NULL)
18	4	(NULL)	7	1	1	10	1	(NULL)
19	4	(NULL)	7	1	1	11	2	(NULL)
20	4	(NULL)	8	2	2	12	1	(NULL)

Рисунок 4.13 – Таблиця фактів FactData

У третій функції FunctionAddForecastData для додавання до сховища записів про прогнозовану генерацію енергії для кожного приладу на локації та запис про прогнозоване споживання електроенергії на локації створюються тимчасові таблиці «temp_devices» та «temp_location» (Рисунки 4.14-4.15).

ForecastData (6r × 13c)

add_date_time	generation	date_time_id	wind_speed	wind_deg	clouds	temperature	sunrise	sunset	weather	location_id	device_id	customer_id
2010-10-31 21:00:00	29	1	12	50	50	4	1 637 125 231	1 637 157 056	1	1	1	1
2010-10-31 21:00:00	45	1	12	50	50	4	1 637 125 231	1 637 157 056	1	1	2	1
2010-10-31 21:00:00	14	1	20,1	77	74	7	1 637 125 231	1 637 157 056	2	2	1	2
2010-11-01 00:00:00	37	4	14,8	93	96	-3	1 637 125 252	1 637 157 015	3	1	1	1
2010-11-01 00:00:00	24	4	14,8	93	96	-3	1 637 125 252	1 637 157 015	3	1	2	1
2010-11-01 00:00:00	41	4	23,4	56	10	6	1 637 125 252	1 637 157 015	4	2	1	2

Рисунок 4.14 – Таблиця temp_devices у функції FunctionAddActualDeviceData

ForecastConsumption (4r × 7c)

date_	weather	consumption	location_id	add_date_weather	date_time_id	customer_id
2010-11-01 00:00:00	1	27,8	1	2010-10-31 21:00:00	1	1
2010-11-01 00:00:00	2	40,8	2	2010-10-31 21:00:00	1	2
2010-11-01 03:00:00	3	24,5	1	2010-11-01 00:00:00	4	1
2010-11-01 03:00:00	4	23,5	2	2010-11-01 00:00:00	4	2

Рисунок 4.15 – Таблиця temp_location у функції FunctionAddActualDeviceData

На рисунках 4.16-4.17 показано роботу третьої функції та формат даних, які записуються до вимірів DimForecastData, DimWeather та DimData та таблиці фактів FactData.

dw_belka.dimforecastdata: 4 строк (приблизите

id	forecast_consumption
1	27,8
2	40,8
3	24,5
4	23,5

dimweather (4r × 8c)

id	add_date_time	wind_speed	wind_deg	clouds	temperature	sunrise	sunset
9	2010-10-31 21:00:00	12	50	50	4	1 637 125 231	1 637 157 056
10	2010-10-31 21:00:00	20,1	77	74	7	1 637 125 231	1 637 157 056
11	2010-11-01 00:00:00	14,8	93	96	-3	1 637 125 252	1 637 157 015
12	2010-11-01 00:00:00	23,4	56	10	6	1 637 125 252	1 637 157 015

Рисунок 4.16 – Виміри

DimData (6r × 4c)

id	generation	state	convector
13	29	1	1
14	45	1	1
15	14	1	1
16	37	1	1
17	24	1	1
18	41	1	1

factdata (10r × 9c)

id	date_time...	location_data...	weather...	customer...	location...	data...	device...	forecast_consumption...
21	1	(NULL)	9	1	1	13	1	(NULL)
22	1	(NULL)	9	1	1	14	2	(NULL)
23	1	(NULL)	9	1	1	(NULL)	(NULL)	1
24	1	(NULL)	10	2	2	15	1	(NULL)
25	1	(NULL)	10	2	2	(NULL)	(NULL)	2
26	4	(NULL)	11	1	1	16	1	(NULL)
27	4	(NULL)	11	1	1	17	2	(NULL)
28	4	(NULL)	11	1	1	(NULL)	(NULL)	3
29	4	(NULL)	12	2	2	18	1	(NULL)
30	4	(NULL)	12	2	2	(NULL)	(NULL)	4

Рисунок 4.17 – Вимір і таблиця фактів

При реалізації функцій було використано оператор «FOR», який був доданий до MariaDB у версії 10.3, тому для роботи зі сховищем потрібна версія не нижче вказаної.

На рисунках 4.18-4.19 наведено приклади роботи запитів зі сховищем даних для отримання актуальних даних, таких що були отримані у певний момент часу з приладів, датчиків чи лічильників на локації. Також, у наведених прикладах можна робити вибірки даних за часом використовуючи оператор «BETWEEN».

```
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_),
'%Y-%m-%d %H:%i:00') as date_, dimlocationdata.accumulator_capacity, dimlocationdata.consumption, dimcustomer.name,
dimcustomer.surname, dimlocation.name, dimdevices.name FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimlocationdata ON dimlocationdata.id = factdata.location_data_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND location_data_id IS NOT NULL;
```

Результат #1 (4r × 7c)

date_	accumulator_capacity	consumption	name	surname	name	name
2010-11-01 00:00:00	85	54,6	Іван	Іваненко	Заправна станція	Акумуляторна батарея Powerwall
2010-11-01 01:00:00	100	46	Іван	Іваненко	Заправна станція	Акумуляторна батарея Powerwall
2010-11-01 02:00:00	75	62,1	Іван	Іваненко	Заправна станція	Акумуляторна батарея Powerwall
2010-11-01 03:00:00	89	61,8	Іван	Іваненко	Заправна станція	Акумуляторна батарея Powerwall

Рисунок 4.18 – Актуальні дані з датчиків першої локації

```
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00') as date_,
dimcustomer.name, dimcustomer.surname, dimlocation.name, dimdevices.name, dimdata.generation, dimweather.clouds, dimweather.wind_speed
FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimdata ON dimdata.id = factdata.data_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND data_id IS NOT NULL AND dimweather.add_date_time =
DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00');
```

Результат #1 (8r × 8c)

date_	name	surname	name	name	generation	clouds	wind_speed
2010-11-01 00:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	27	61	14,4
2010-11-01 01:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	38	72	22,4
2010-11-01 02:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	30	38	4
2010-11-01 03:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	29	97	16,6
2010-11-01 00:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор...	13	61	14,4
2010-11-01 01:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор...	25	72	22,4
2010-11-01 02:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор...	15	38	4
2010-11-01 03:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор...	12	97	16,6

Рисунок 4.19 – Актуальні дані генерації приладами енергії для першої локації

На рисунках 4.20-4.21 наведено приклади роботи запитів зі сховищем даних для отримання прогнозованих даних.

```
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00') as date_,
dimcustomer.name, dimcustomer.surname, dimlocation.name, dimdevices.name, dimdata.generation, dimweather.clouds, dimweather.wind_speed
FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimdata ON dimdata.id = factdata.data_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND data_id IS NOT NULL AND dimweather.add_date_time !=
DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00');
```

Результат #1 (4r × 8c)

date_	name	surname	name	name	generation	clouds	wind_speed
2010-11-01 00:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	29	50	12
2010-11-01 03:00:00	Іван	Іваненко	Заправна станція	TSM-DE19-540M	37	96	14,8
2010-11-01 00:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор з контрол...	45	50	12
2010-11-01 03:00:00	Іван	Іваненко	Заправна станція	Вітряний генератор з контрол...	24	96	14,8

Рисунок 4.20 – Прогнозована генерація для приладів першої локації

```
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_),
 '%Y-%m-%d %H:%i:00') as date_, dimcustomer.name, dimcustomer.surname, dimlocation.name, dimweather.clouds,
dimweather.wind_speed FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimforecastdata ON dimforecastdata.id = factdata.forecast_consumption_id
WHERE customer_id = 1 AND location_id = 1 AND forecast_consumption_id IS NOT NULL;
```

Результат #1 (2r × 6c)

date_	name	surname	name	clouds	wind_speed
2010-11-01 00:00:00	Іван	Іваненко	Заправна станція	50	12
2010-11-01 03:00:00	Іван	Іваненко	Заправна станція	96	14,8

Рисунок 4.21 – Прогнозоване споживання електроенергії локацією

Для роботи з вітринами достатньо скористатися виразом «Create View» та обрати всі необхідні дані для виду (View). На рисунку 4.22 наведено приклад створення виду з даними про користувача з таблиці бази даних.

```
CREATE VIEW CustomerView AS SELECT name, surname, login, PASSWORD, email, phone
FROM Customer WHERE id = 1;

SELECT * FROM customerview;
```

customerview (1r × 6c)

name	surname	login	PASSWORD	email	phone
Іван	Іваненко	ivan_ivan	12345	ivan.ivaenko@gmail.com	+380669241145

Рисунок 4.22 – Робота з видами у базі

На рисунку 4.23 наведено приклад створення виду з даними прогнозованого споживання електроенергії у локації зі сховища даних.

```
CREATE VIEW CustomerForecastLocationData AS
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day, ' ', dimdatetime.time_),
 '%Y-%m-%d %H:%i:00') as date_, dimweather.clouds, dimweather.wind_speed, dimweather.temperature, dimweather.sunrise,
dimweather.sunset, dimforecastdata.forecast_consumption FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimforecastdata ON dimforecastdata.id = factdata.forecast_consumption_id
WHERE customer_id = 1 AND location_id = 1 AND forecast_consumption_id IS NOT NULL;


SELECT * FROM CustomerForecastLocationData;
```

CustomerForecastLocationData (2r × 7c)

date_	clouds	wind_speed	temperature	sunrise	sunset	forecast_consumption
2010-11-01 00:00:00	50	12	4	1 637 125 231	1 637 157 056	27,8
2010-11-01 03:00:00	96	14,8	-3	1 637 125 252	1 637 157 015	24,5

Рисунок 4.23 – Робота з видами у сховищі

Для архівування даних зі сховища можна використовувати .CSV файл. Неможливо створити скрипт автоматичної архівації використовуючи лише мову структурованих запитів (SQL). Тому існує декілька способів вирішення цього завдання: архівувати дані вручну, або автоматизувати цей процес використавши PHP. У додатку В наведено програмний код PHP функції для збереження даних будь-якого масиву в .CSV файл. Для збереження даних необхідно зробити вибірку зі сховища та просто передати отриманий масив у функцію export_to_csv. Результати роботи функції наведено на рисунку 4.24.

	A	B	C	D	E	F	G	H	I
1	id	date_time_id	location_data_id	weather_id	customer_	location_id	data_id	device_id	forecast_consumption_id
2	1	1	1	\N	1	1	\N	3	\N
3	2	1	2	\N	2	2	\N	4	\N
4	3	2	3	\N	1	1	\N	3	\N
5	4	2	4	\N	2	2	\N	4	\N
6	5	3	5	\N	1	1	\N	3	\N
7	6	3	6	\N	2	2	\N	4	\N
8	7	4	7	\N	1	1	\N	3	\N
9	8	4	8	\N	2	2	\N	4	\N
10	9	1	\N	1	1	1	1	1	\N
11	10	1	\N	1	1	1	2	2	\N
12	11	1	\N	2	2	2	3	1	\N

Рисунок 4.24 – Результати роботи функції експорту

4.2 Розгортання у хмарному середовищі

Для розгортання у хмарному середовищі було обрано хмарну платформу Amazon Web Services (AWS) та веб-сервіс Amazon Relational Database Service (Amazon RDS) для роботи з реляційними базами даних у хмарі. RDS – вирізняється гнучким масштабуванням та економічністю емності [24]. Для початку роботи з AWS потрібно створити акаунт, реєстрація проходить у п’ять етапів (Рисунок 4.25).

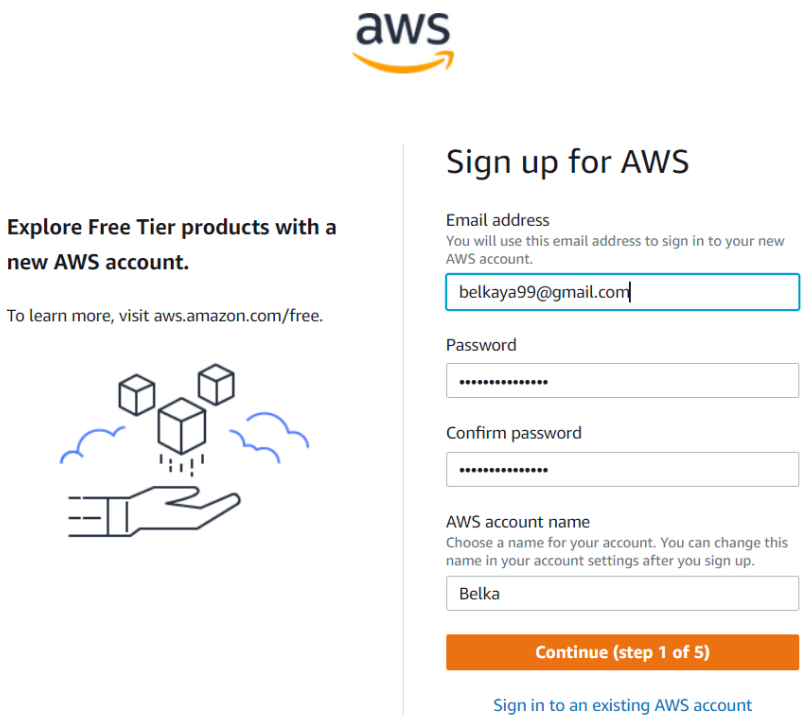


Рисунок 4.25 – Крок перший створення акаунту AWS

На другому кроці потрібно ввести свої контактні дані: ім’я, телефон, адресу проживання та обрати план використання сервісів Amazon – бізнес або персональний. Третій крок – це введення даних банківської карти, а четвертий крок – підтвердження особистості через номер телефону. На заключному п’ятому етапі необхідно обрати план підтримки (Рисунок 4.26).

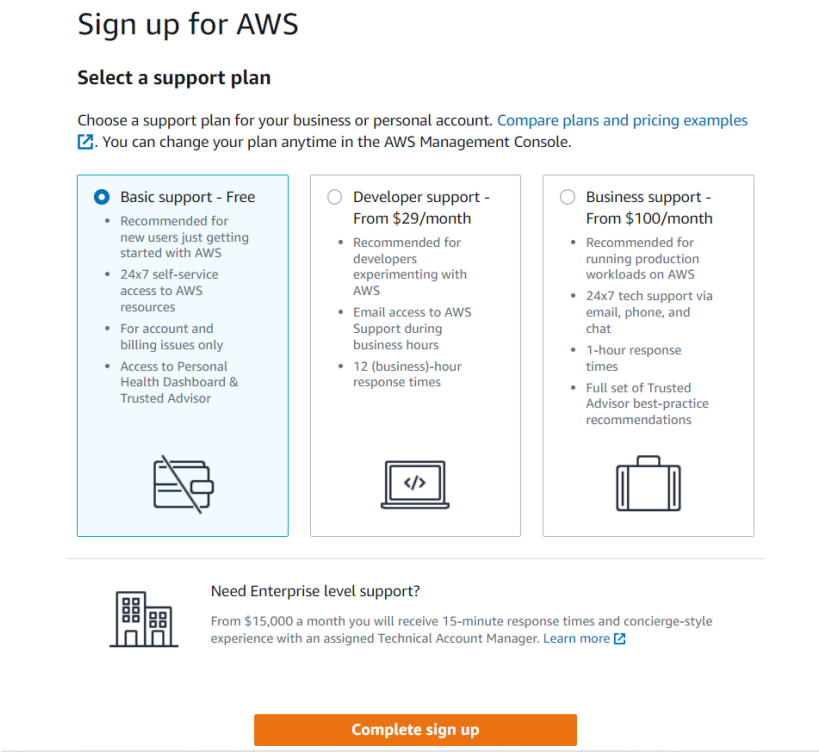


Рисунок 4.26 – Заключний крок створення акаунту AWS

Для демонстрації процесу розгортання бази даних у хмарі було обрано безкоштовний план. Після обрання плану з’являється повідомлення про завершення реєстрації акаунту та попередження, що його активування може зайняти кілька хвилин (Рисунок 4.27). Після завершення активації на вказану при реєстрації пошту приходить лист.

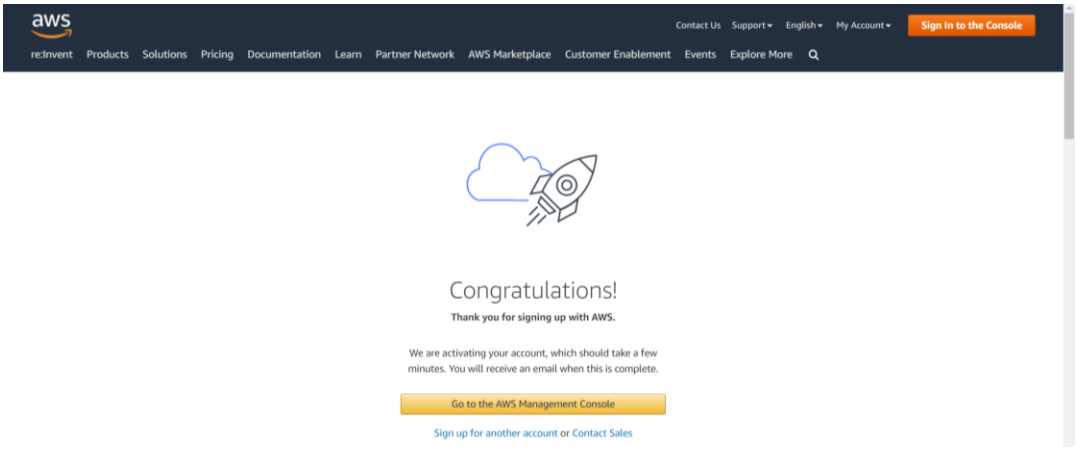


Рисунок 4.27 – Заключний крок створення акаунту AWS

Після активації акаунту потрібно перейти до пункту меню RDS через Services (Сервіси) – Database (База даних) – RDS (Рисунок 4.28).

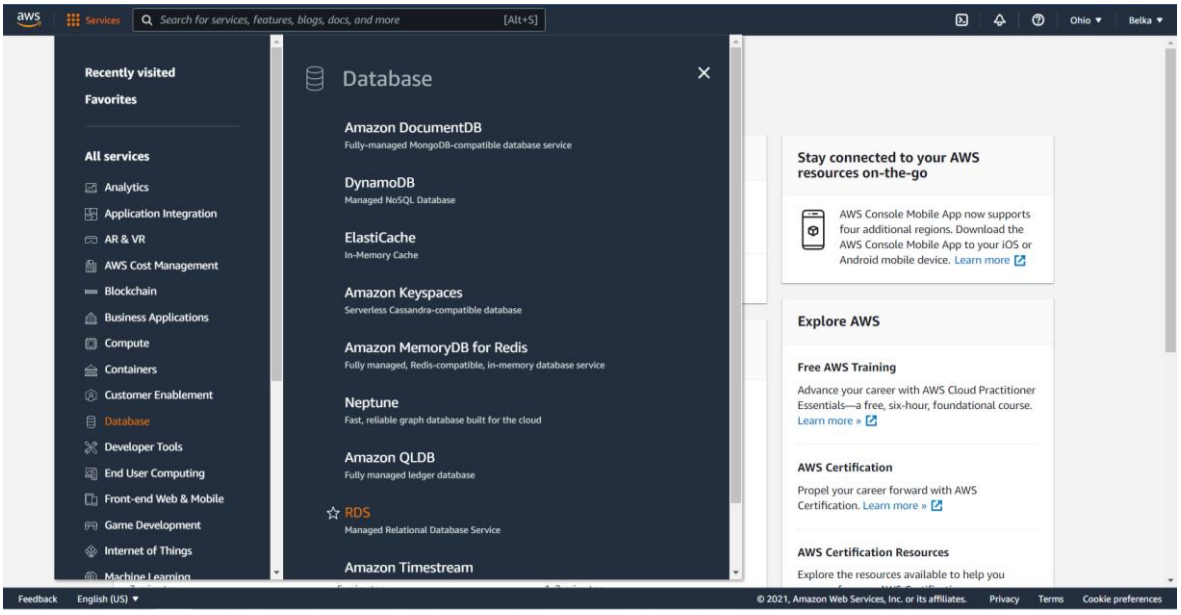


Рисунок 4.28 – Перехід до меню створення нової бази даних

Після натискання опції «Create database» з’являється форма для заповнення з параметрами бази на вибір (Рисунок 4.29).

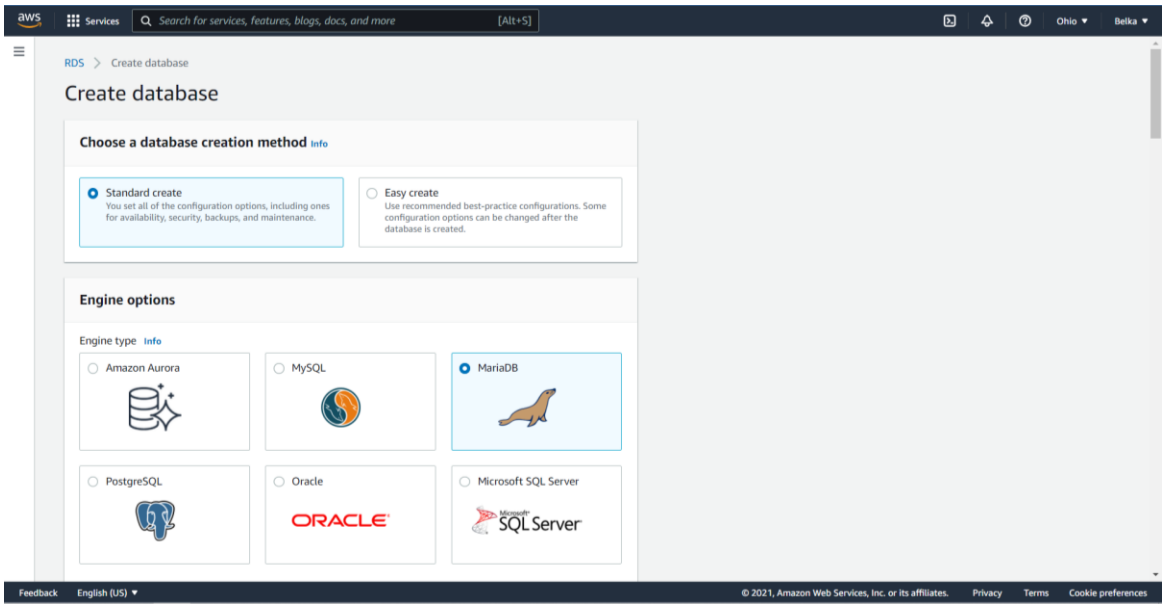


Рисунок 4.29 – Створення бази даних

Необхідно обрати систему керування базами даних та її версію. RDS пропонує на вибір Amazon Aurora, MySQl, PostgreSQL, Oracle та Microsoft SQL Server. Разом з системою потрібно обрати її версію. Далі в налаштуваннях зазначається назва бази та параметри для входу (ім’я та пароль) головного користувача. Серед параметрів сховища можна обрати розмір від двадцяти до більш ніж шістнадцяти тисяч гібібайт (GiB). Різниця гігабайту (GB) та гігібайту GiB у системі обчислень 1GB дорівнює 10^9 байт, а 1 GiB це 2^{30} байт (Рисунок 4.30).

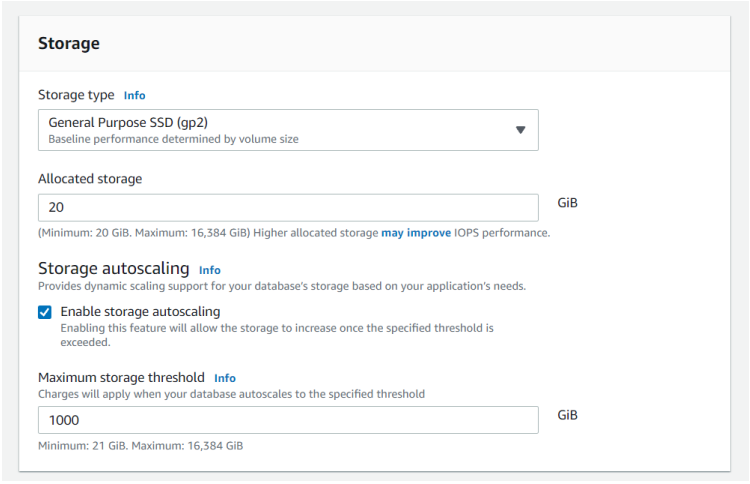


Рисунок 4.30 – Вибір параметрів сховища

Ще одним корисним налаштуванням є вибір параметрів резервного копіювання, у період від одного до тридцяти п’яти днів (Рисунок 4.31).

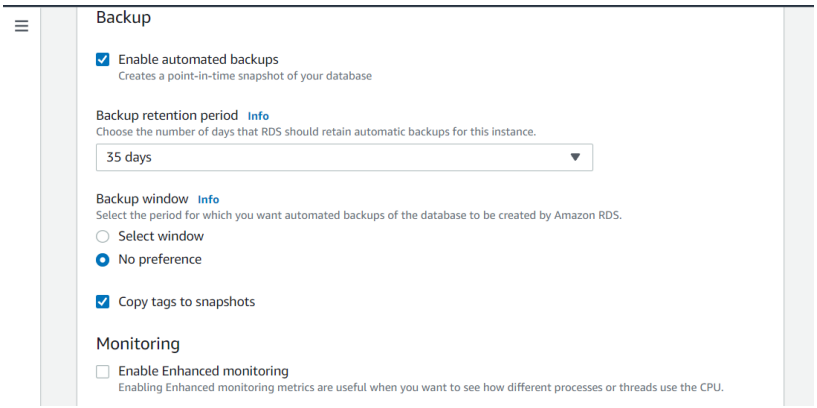


Рисунок 4.31 – Вибір параметрів резервного копіювання

Після заповнення параметрів нової бази з’явиться повідомлення з проханням зачекати кілька хвилин до завантаження бази даних (Рисунок 4.32).

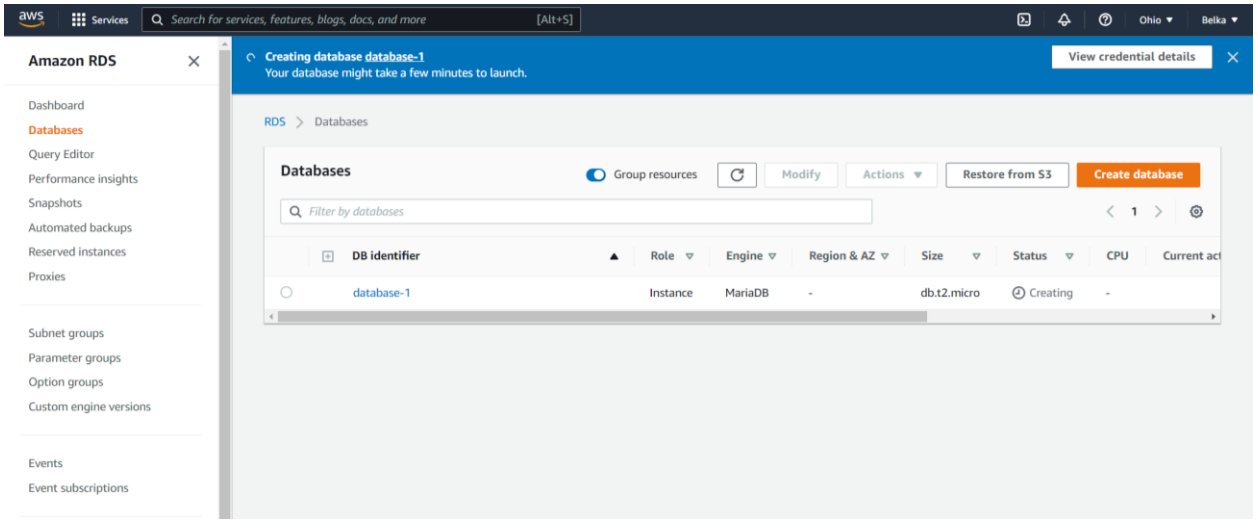


Рисунок 4.32 – Завершення створення бази

Після завершення завантаження бази можна переглянути параметри (Рисунок 4.33). Параметр «endpoint» - кінцева точка знадобиться для подальшого імпорту бази.

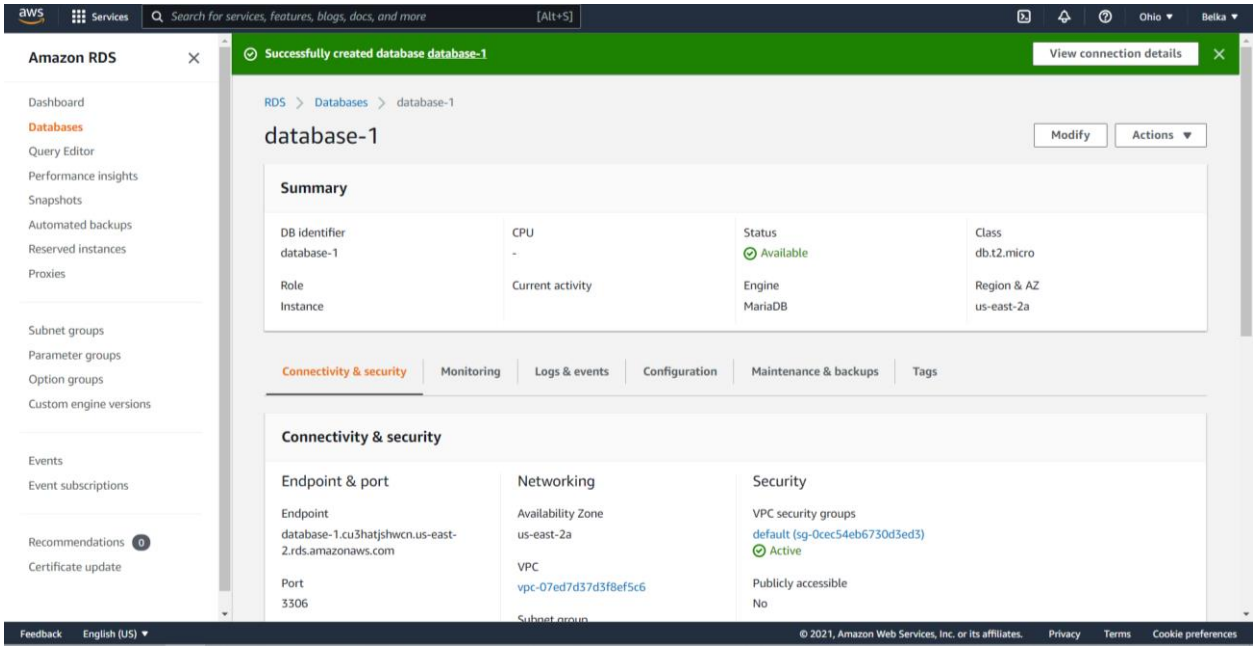


Рисунок 4.33 – Перегляд параметрів

Спочатку потрібно створити з’єднання (Рисунок 4.34). Для цього можна використати будь-який зручний інструмент адміністрування баз даних. У прикладі було використано MySQL Workbench.

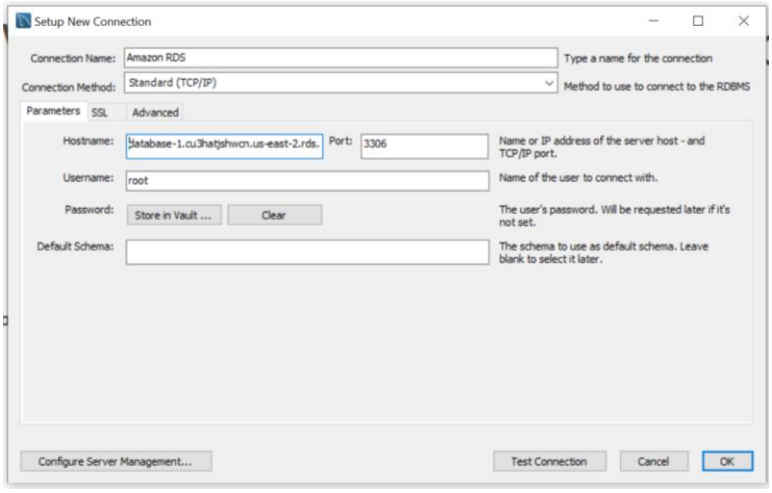


Рисунок 4.34 –Створення нового з’єднання

Після створення з’єднання (Рисунок 4.35) потрібно підключитися використовуючи пароль головного користувача.

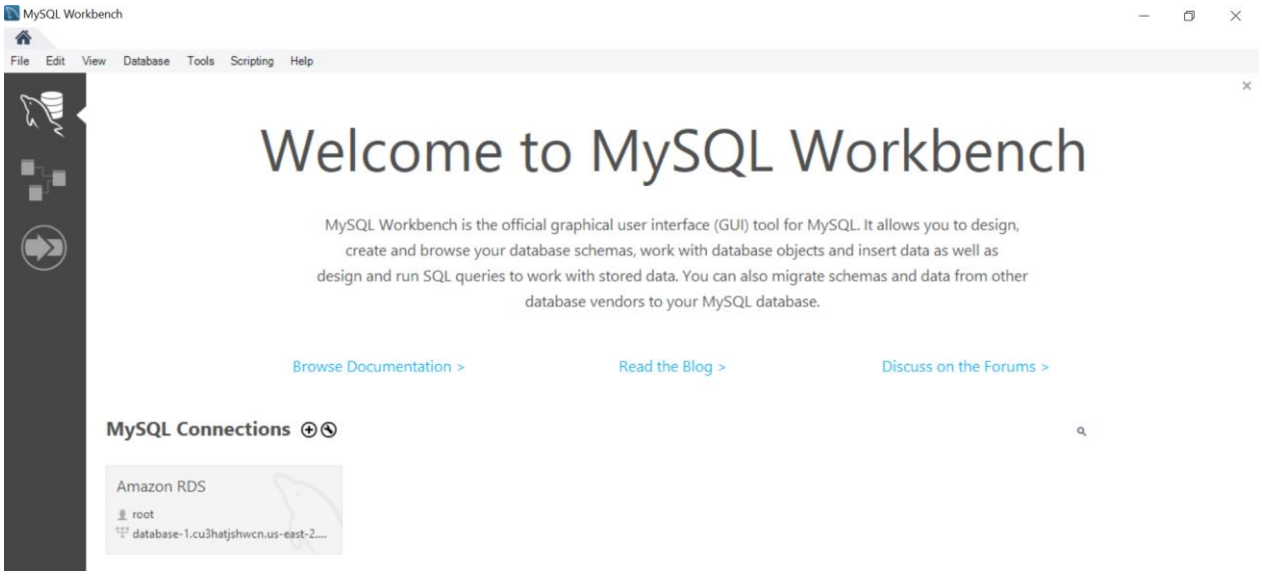


Рисунок 4.35 – Створене з’єднання

Далі залишається лише перейти до Administration-Data Import/Restore та завантажити файл зі скриптами бази даних (Рисунок 4.36).

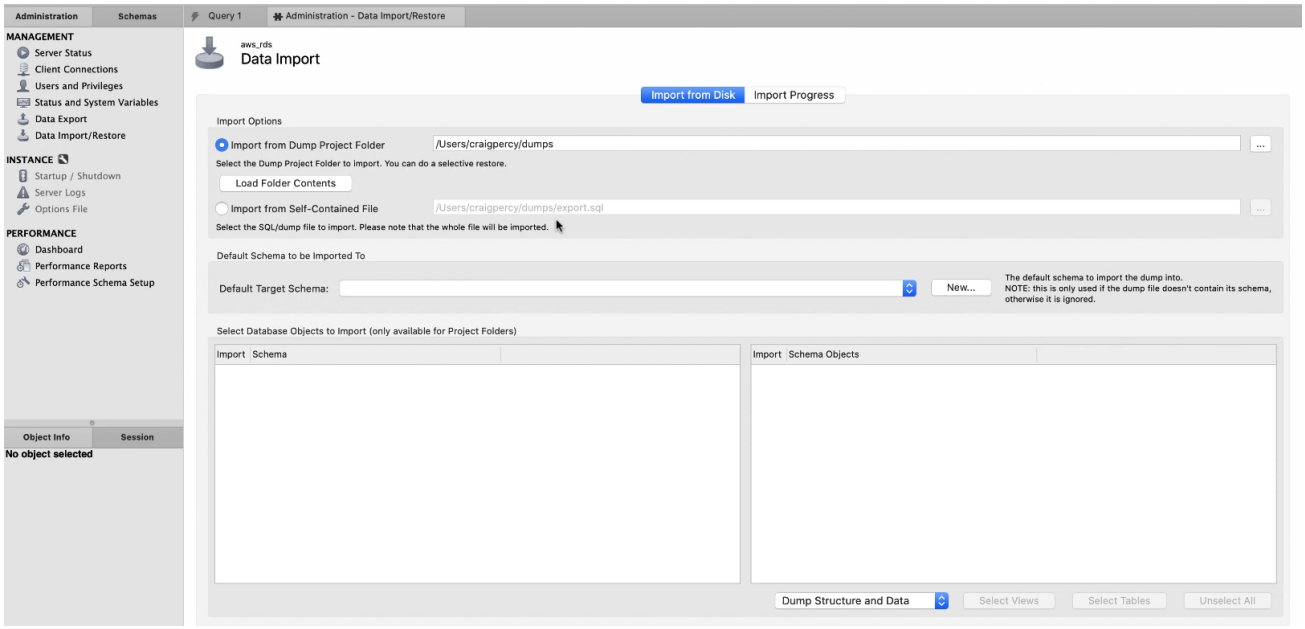


Рисунок 4.36 – Процес імпорту бази

Після цього, сховище готове до використання з усіма перевагами Amazon RDS:

- Amazon RDS керує резервним копіюванням, надає послугу автоматичного виявлення збоїв та відновлення у разі їх виникнення.
- Можна або включити автоматичне резервне копіювання для відновлення бази даних або проводити його вручну. Процес відновлення RDS працює надійно та ефективно.
- RDS підтримує автоматичне масштабування та надає великий обсяг місця для роботи.
- Також дані можна захистити за допомогою до віртуальної приватної хмари (VPC).

ВИСНОВКИ

Перехід на використання відновлюваних джерел енергії є важливим кроком у зменшенні кількості викидів парникових газів в атмосферу. Вітряні електроустановки та сонячні батареї є доступними для використання як для промислових так і для побутових споживачів. За мету дослідження було взято покращення ефективності у прийнятті рішень при управлінні енергетичними мікромережами шляхом розробки сховища даних.

У першому розділі було проведено аналіз предметної області, а саме: ідентифіковано проблему та проведено аналіз аналогів. Також було розглянуто та проведено порівняння моделей представлення даних у сховищі при управлінні енергетичною інфраструктурою. Для роботи СППР потрібно обробляти великий масив інформації про життєвий цикл енергії. Для функціонування системи з якою працює користувач було вирішено створити базу операційних даних, а для навчання моделі прогнозування генерації енергії окреме централізоване сховище даних, для того, щоб розділити навантаження аналітики та постійну роботу системи і уникнути затримок у щоденних операціях користувачів.

Аналіз аналогів показав, що попередні дослідники енергосистем використовували модель «Зірки» для реалізації сховища даних, що має свої недоліки через денормалізовану структуру, які усуває використання «Сніжинки», яку і було обрано після аналізу. «Сніжинка» яка має нормалізовану структуру даних, а тому займає менше дискового простору, завдяки чому приводить до зниження надмірності даних та запобіганню втратам пам’яті.

У другому розділі було докладніше описано мету та задачі дослідження, серед яких задачі проектування архітектури сховища та бази даних та розробка процедур та запитів для роботи СППР зі сховищем даних.

У третьому розділі проведено проектування архітектури сховища даних та описано діаграму потоку даних. Під час концептуального моделювання було виділено

двадцять дві сутності операційної бази даних та описано значення їх атрибутів. Побудована логічна модель операційної бази даних та сховища даних відобразила зв’язки між сутностями системи.

У четвертому розділі було розроблено процедуру для завантаження даних з операційної бази до сховища та наведено результати роботи функцій та запитів зі сховищем. Також було проведено розгортання бази у хмарному середовищі AWS.

Отже, розроблене сховище даних розроблене для інтеграції до системи підтримки прийняття рішень при управлінні енергетичними мікромережами за змінних метеорологічних та географічних умов для обробки інформації про життєвий цикл енергії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pavlowsky C. Individual and local scale interactions and adaptations to wind energy development: A case study of Oklahoma, USA, 2021. 7 с.

2. Agbo E. Solar energy: A panacea for the electricity generation crisis in Nigeria, 2021. 21 с.

3. Cousse J. Still in love with solar energy? Installation size, affect, and the social acceptance of renewable energy technologies, 2021. – 14 с.

4. Стимулювання відновлюваної енергетики в Україні за допомогою «зеленого» тарифу, 2012. 80 с. – (Консультативна програма IFC в Європі та Центральній Азії).

5. Cardon D. Database vs. Data Warehouse: A Comparative Review, 2018. 5 с. – (Health Catalyst).

6. Han J., Kamber M., Pei J. Data Mining Consepts and Techniques Third Edition, 2012. 740 с. – (Elsevier).

7. Parekh A. Introduction on Data Warehouse with OLTP and OLAP – Bhavnagar, 2013. 5 с. – (International Journal Of Engineering And Computer Science).

8. Qishan Y. Analysis of Data Warehouse Architectures: Modeling and Classification, 2019. 9 с.

9. Li Y. A Data Warehouse Architecture supporting Energy Management of Intelligent Electricity System, 2013. 4 с.

10. Samsinar R., Endro Suseno J., Edi Widodo C. Power Distribution Analysis For Electrical Usage In Province Area Using Olap (Online Analytical Processing), 2018. 5 с. – (ICENIS). – (E3S Web of Conferences 31).

11. Freschi F. Artificial Immune System in the Management of Complex Small Scale Cogeneration Systems – Torino, 2009. 18 с.

12. Moody D. L., Kortink M. A. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design, 2000. 12 с.

13. Gökçe U. Multi-dimensional energy monitoring, analysis and optimization system for energy efficient building operations, 2014. 13 с.

14. Sarka D., Lah M., Jerkic G. Implementing a Data Warehouse with Microsoft SQL Server 2012, 2014. 816 с. – (Microsoft Official Academic Course (COR) Moac).

15. Levene M., Loizou G. Why is the snowflake schema a good data warehouse design?, 2003. 21 с. – (Elsevier). – (Information Systems).

16. Karmani M., Qureshi J., Sarwar N. A review of Star schema and Snowflakes Schema, 2020. 13 с.

17. Freedman A. DEDICATED vs. CLOUD: Comparing dedicated and cloud infrastructure for high availability (HA) and non-high availability applications, 2018. 17 с.

18. Шендрик С. О. Моделі та інформаційна технологія підтримки прийняття рішень при управлінні гібридними енергомережами: дисертація доктора філософії : 122 Комп'ютерні науки – Харків, 2020. 206 с.

19. Важинський С.Е., Щербак Т І. Методика та організація наукових досліджень, – Суми, 2016. 260 с.

20. Bielik L. Methodology of science An introduction – Bratislava: Comenius University in Bratislava, 2019. 227 с.

21. Davis W. S., Yen D. C. The Information System Consultant's Handbook, 1998. 800 с.

22. Sparx Systems. From Conceptual Model to DBMS / Sparx Systems., 2017. 18 с. – (Data Modeling).

23. Kromann F. Beginning PHP and MySQL From Novice to Professional Fifth Edition – Aliso Viejo, CA, USA, 2018. 889 с.

24. Amazon Relational Database Service User Guide, 2021. – (Amazon Web Services).

25. A guide to the Project Management Body of Knowledge (PMBOK Guide) – Newtown Square, Pennsylvania: Project Management Institute, Inc., 2018. 592 с. – (Sixth Edition).

26. Appigatla K. MySQL 8 Cookbook: Over 150 recipes for high-performance database querying and administration, 2018. 969 с.

27. Burge S. MySQL Explained: Your Step By Step Guide to Database Design, 2017. 390 с.

28. Schallehn E. Advanced Database Models – Magdeburg, 2019. 358 с.

29. Bouaziz S., Nabli A., Gargouri F. Design a data warehouse schema from document-oriented database, 2019. – 10 с. – (Procedia Computer Science).

30. Bhatia P. Data Mining and Data Warehousing Principles and Practical Techniques, 2019. 506 с.

ДОДАТОК А

АНАЛІЗ ПЛАНУВАННЯ РОБІТ

Ідентифікація мети ІТ-проекту. Продуктом дипломного проекту є «Сховище даних для системи підтримки прийняття рішень при управлінні енергетичними мікромережами», призначений для підвищення ефективності у прийнятті рішень при управлінні енергетичними мікромережами. Перед початком виконання проекту було проведено деталізацію мети методом SMART. Результати деталізації розміщені у табл. А.1.

Таблиця А.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити сховище даних та розгорнути його на хмарній платформі для використання системою підтримки прийняття рішень при управлінні енергетичними мікромережами.
Measurable (вимірювана)	Головним критерієм результату є оцінка замовника.
Achievable (досяжна)	Мета є досяжною, оскільки розробник має необхідні навички та знання для розробки сховища даних.
Relevant (реалістична)	Для розробки сховища даних, достатньо кваліфікований розробник має усі необхідні програми, технічні засоби та доступ до Інтернет-мережі.
Time-framed (обмежена у часі)	Сховище даних створюється у відповідності до дат, зазначених у календарному плані.

Планування змісту структури робіт IT-проекту. Планування змісту структури робіт здійснено за допомогою WBS діаграми, яка графічно представляє елементи проекту у вигляді пакетів робіт. Структура декомпозиції робіт є інструментом розбиття великих задач на менші, що допомагає краще управляти проектом. Діаграма WBS зображена на рис. А.1.

Планування структури організації, для впровадження готового проекту (OBS). Організаційна структура виконавців OBS забезпечує призначення команд чи окремих виконавців для виконання кожної задачі. Діаграма OBS представлена на рис. А.2. Список виконавців в проекті наведений в табл. А.2.

Таблиця А.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник, тестувальник	Бєлка Я.С.	Виконує проектування та розробку сховища даних. Виконує збір та аналіз даних.
Керівник проекту	Шендрик В.В.,	Формує технічне завдання на розробку проекту. Затверджує модель бази. Відповідає за контроль термінів.

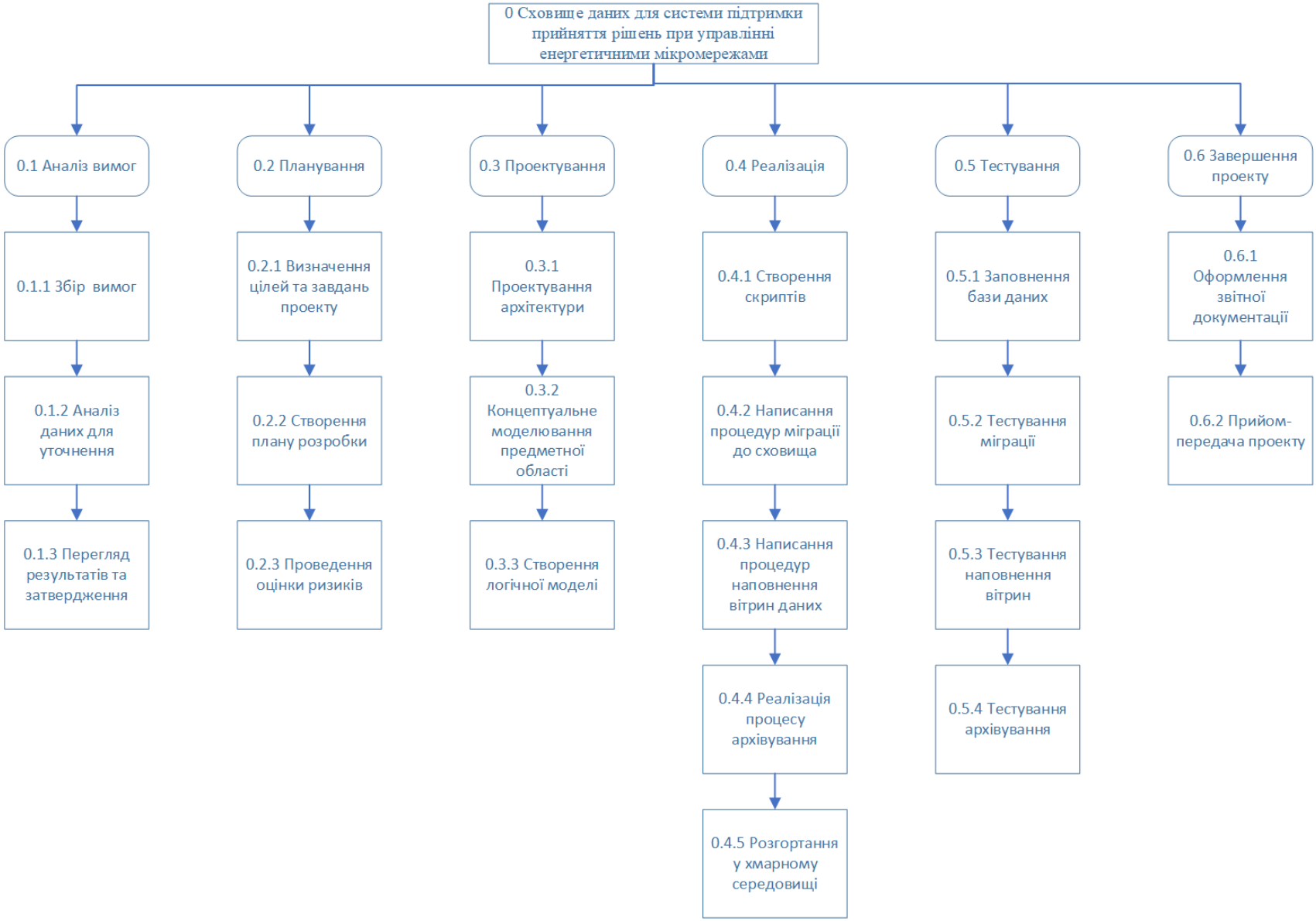


Рисунок А.1 − WBS Структура робіт проекту

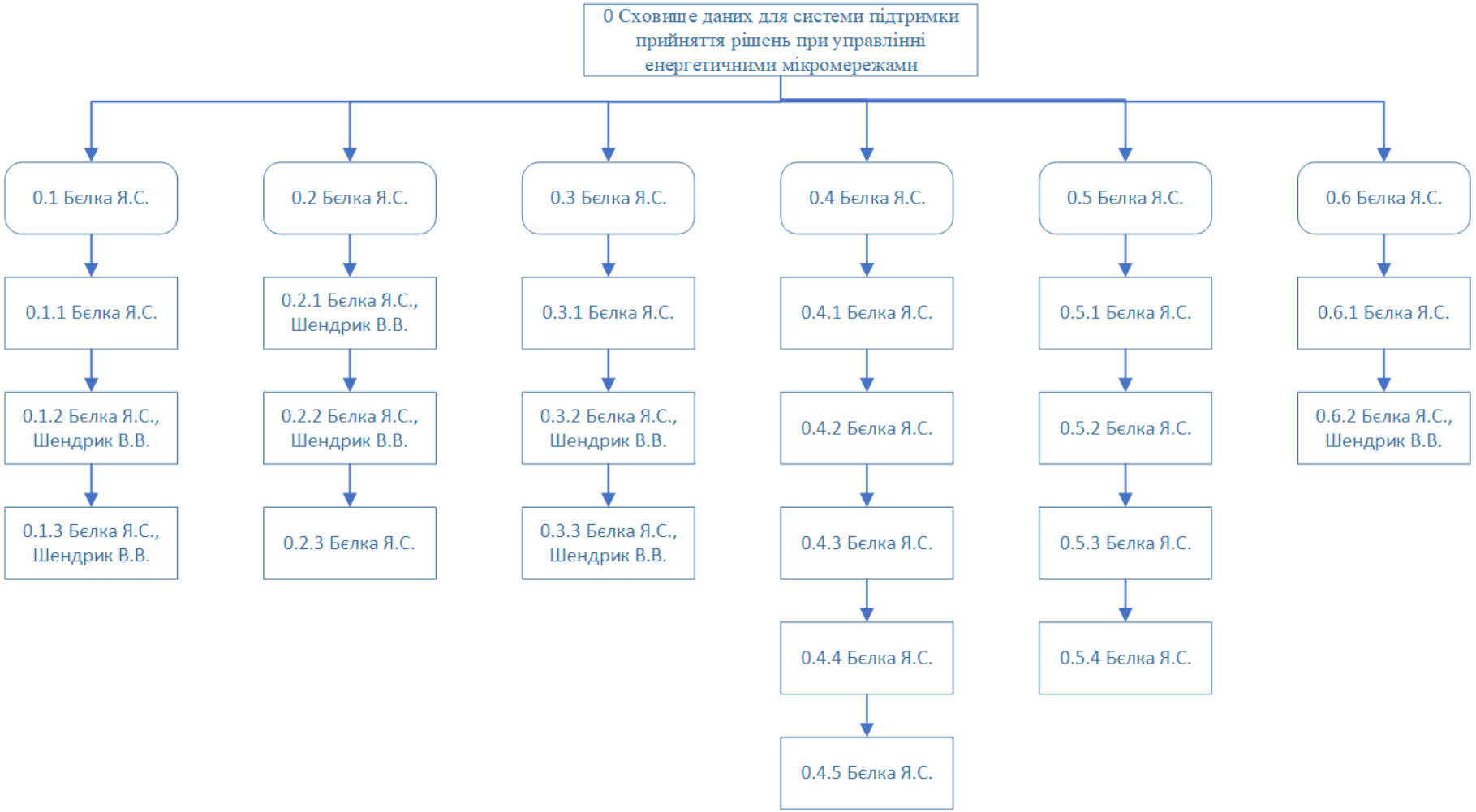


Рисунок А.2 − Організаційна структура проекту (OBS)

Побудова календарного графіку виконання IT-проекту. Календарний план виконання проекту виконаний у формі діаграми Ганта. Діаграма візуалізує графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт зазначена в днях. Фактична тривалість виконання робіт дорівнює 6 годинам на день. Діаграма Ганта та список робіт зображені на рис. А.3-А.4.

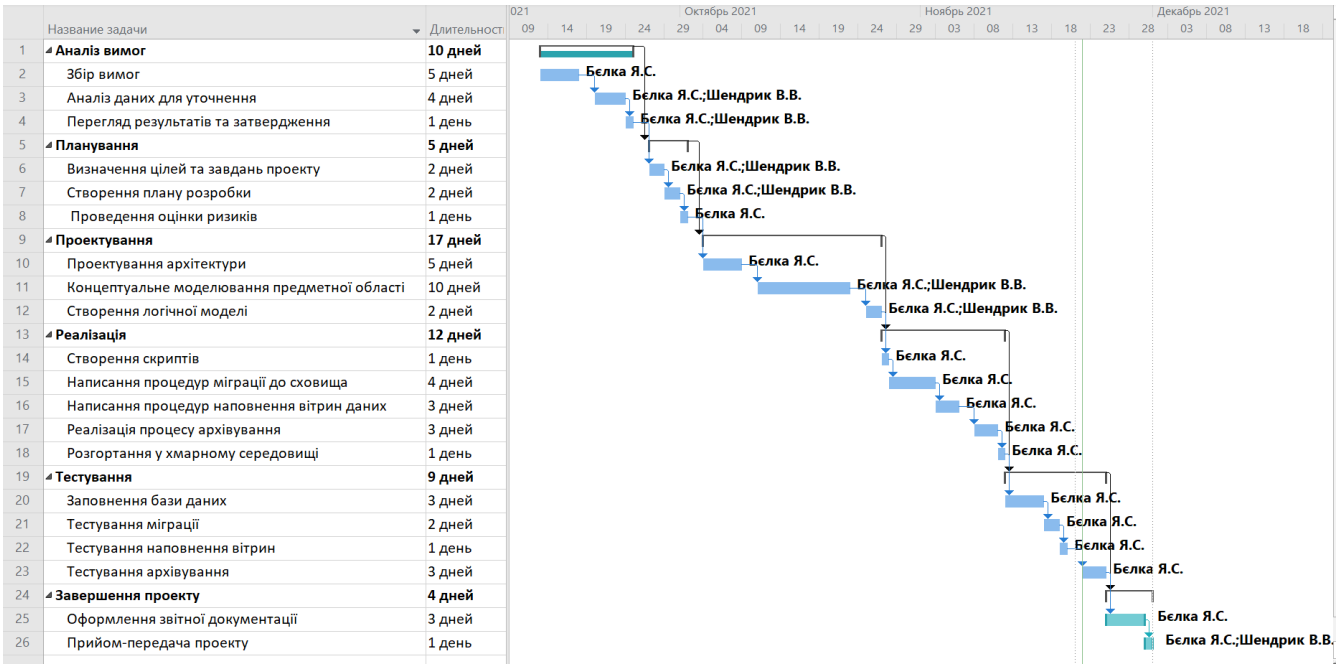


Рисунок А.3 − Діаграма Ганта

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	**Аналіз вимог**	**10 дней**	**Пн 13.09.21**	**Пт 24.09.21**		**Бєлка Я.С.**
2	Збір вимог	5 дней	Пн 13.09.21	Пт 17.09.21		Бєлка Я.С.
3	Аналіз даних для уточнення	4 дней	Пн 20.09.21	Чт 23.09.21	2	Бєлка Я.С.;Шендрик В.В.
4	Перегляд результатів та затвердження	1 день	Пт 24.09.21	Пт 24.09.21	3	Бєлка Я.С.;Шендрик В.В.
5	**Планування**	**5 дней**	**Пн 27.09.21**	**Пт 01.10.21**	1	**Бєлка Я.С.**
6	Визначення цілей та завдань проекту	2 дней	Пн 27.09.21	Вт 28.09.21	4	Бєлка Я.С.;Шендрик В.В.
7	Створення плану розробки	2 дней	Ср 29.09.21	Чт 30.09.21	6	Бєлка Я.С.;Шендрик В.В.
8	Проведення оцінки ризиків	1 день	Пт 01.10.21	Пт 01.10.21	7	Бєлка Я.С.
9	**Проектування**	**17 дней**	**Пн 04.10.21**	**Вт 26.10.21**	5	**Бєлка Я.С.;Шендрик В.В.**
10	Проектування архітектури	5 дней	Пн 04.10.21	Пт 08.10.21	8	Бєлка Я.С.
11	Концептуальне моделювання предметної області	10 дней	Пн 11.10.21	Пт 22.10.21	10	Бєлка Я.С.;Шендрик В.В.
12	Створення логічної моделі	2 дней	Пн 25.10.21	Вт 26.10.21	11	Бєлка Я.С.;Шендрик В.В.
13	**Реалізація**	**12 дней**	**Ср 27.10.21**	**Чт 11.11.21**	**9**	**Бєлка Я.С.**
14	Створення скриптів	1 день	Ср 27.10.21	Ср 27.10.21	12	Бєлка Я.С.
15	Написання процедур міграції до сховища	4 дней	Чт 28.10.21	Вт 02.11.21	14	Бєлка Я.С.
16	Написання процедур наповнення вітрин даних	3 дней	Ср 03.11.21	Пт 05.11.21	15	Бєлка Я.С.
17	Реалізація процесу архівування	3 дней	Пн 08.11.21	Ср 10.11.21	16	Бєлка Я.С.
18	Розгортання у хмарному середовищі	1 день	Чт 11.11.21	Чт 11.11.21	17	Бєлка Я.С.
19	**Тестування**	**9 дней**	**Пт 12.11.21**	**Ср 24.11.21**	**13**	**Бєлка Я.С.**
20	Заповнення бази даних	3 дней	Пт 12.11.21	Вт 16.11.21	18	Бєлка Я.С.
21	Тестування міграції	2 дней	Ср 17.11.21	Чт 18.11.21	20	Бєлка Я.С.
22	Тестування наповнення вітрин	1 день	Пт 19.11.21	Пт 19.11.21	21	Бєлка Я.С.
23	Тестування архівування	3 дней	Пн 22.11.21	Ср 24.11.21	22	Бєлка Я.С.
24	**Завершення проекту**	**4 дней**	**Чт 25.11.21**	**Вт 30.11.21**	**19**	**Бєлка Я.С.**
25	Оформлення звітної документації	3 дней	Чт 25.11.21	Пн 29.11.21	23	Бєлка Я.С.
26	Прийом-передача проекту	1 день	Вт 30.11.21	Вт 30.11.21	25	Бєлка Я.С.;Шендрик В.В.

Рисунок А.4 − Список робіт для побудови діаграми Ганта

Планування ризиків проекту. Аналіз ризиків розпочнемо з якісної оцінки, спочатку визначимо пріоритет ризиків для подальшого аналізу, а потім їх вірогідність виникнення та вплив. Це допоможе зменшити неоднозначність і зосередити увагу на пріоритетних ризиках. Після якісної оцінки проведемо кількісну – чисельний аналіз впливу виділених ризиків на цілі проекту, де ризику присвоюється числовий рейтинг. У табл. А.5 наведена класифікація ризиків за показниками ймовірності виникнення ризику та величиною втрат.

Для зведення негативного впливу загроз на цілі проекту до мінімуму виконаємо планування реагування на ризики. Планування враховує ризики за їх пріоритетом, впливом на бюджет, графік. Ризики оцінюються за показниками, що наведені в табл. А.3. На основі оцінки побудовано матрицю ймовірності виникнення та впливу ризиків, що зображена на рис. А.5.

Таблиця А.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

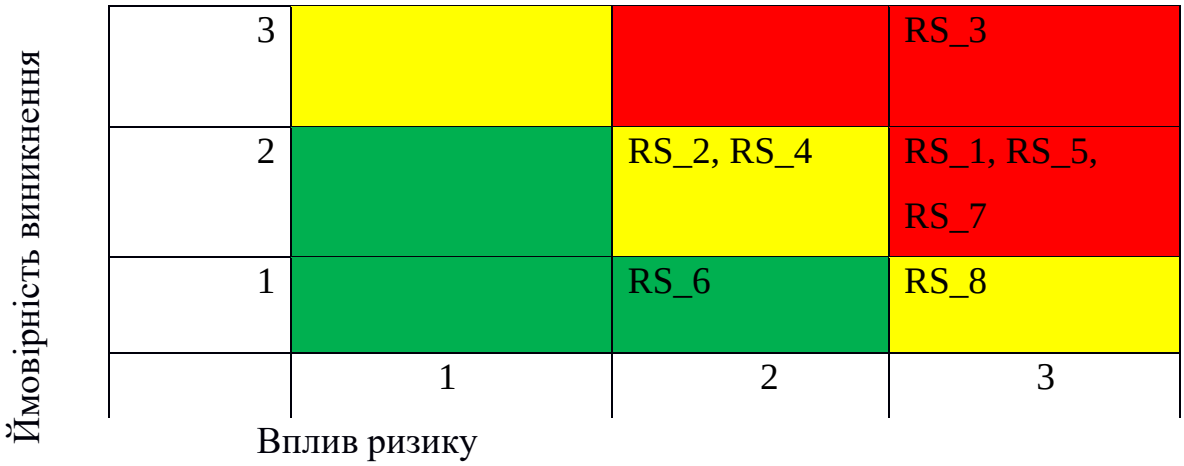


Рисунок А.5 − Матриця ймовірності виникнення ризиків та впливу ризику

– зелений колір – прийнятні ризики;
– жовтий колір – виправданні ризики;
– червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують за рівнем, що знаходиться в табл. А.4.

Таблиця А.4− Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	$1 \leq R \leq 2$	6
2	Виправдані	$3 \leq R \leq 4$	2, 4, 8
3	Недопустимі	$6 \leq R \leq 9$	1, 3, 5, 7

Таблиця А.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус	Опис ризику	Ймовірність виникнення	Вплив	Ранг	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Нечітке технічне завдання	Середня	Високий	6	Перед початком виконання обговорити усі незрозумілості. Співпрацювати з замовником на усіх етапах роботи.	Попередження	При виявленні невідповідностей потрібно негайно оцінити межі впливу похибки та виправити її.
RS_2	Відкритий	Недостатня кваліфікація розробника	Середня	Середній	4	Використання наявних онлайн-ресурсів для підвищення кваліфікації.	Пом’якшення	Використовувати тематичну літературу.
RS_3	Відкритий	Неоптимальний розподіл часу	Висока	Високий	9	Створити структуру робіт і виділяти час виконання на найменші пакети, щоб зменшити можливість похибок розподілу часу. Дотримуватися термінів, визначених у календарному плані.	Пом’якшення	Оптимізувати роботу згідно створеному плану. Розглянути можливість внесення правок до термінів виконання.

Продовження таблиці А.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив	Ранг	План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Часте внесення змін у ТЗ	Середня	Середній	4	Перед початком виконання узгодити з замовником можливість внесення змін та окреслити їх.	Попередження	Діяти за потребою, розглядаючи можливість та варіанти внесення змін.
RS_5	Відкритий	Вибір недоцільної технології розробки	Середня	Високий	6	Проаналізувати наявні аналоги та технології виконання проекту.	Пом’якшення	Приділити час на ознайомлення з технологією, мінімізацію її слабких сторін.
RS_6	Відкритий	Недооцінений масштаб проекту	Низька	Середній	3	На етапі планування провести аналіз проекту. Визначити структуру робіт.	Пом’якшення	Перебудова стратегії реалізації проекту.
RS_7	Відкритий	Не правильно створена архітектура проекту	Середній	Високий	6	Детально продумати та створити діаграми структури проекту на етапі проектування.	Пом’якшення	Моніторити процес реалізації для швидко реагування на несумісності.
RS_8	Відкритий	Відсутність резервних копій	Низька	Високий	3	Вибір хмарного середовища з урахуванням послуг здійснення резервних копій.	Попередження	Налаштувати збереження даних на стороні провайдера.

ДОДАТОК Б

Лістинг створення таблиць бази та сховища даних

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

-- Структура таблиці `Accumulator`

CREATE TABLE `Accumulator` (
  `id` int(11) NOT NULL,
  `location_id` int(11) NOT NULL,
  `device_id` int(11) NOT NULL,
  `charge` tinyint(4) NOT NULL,
  `discharge` tinyint(4) NOT NULL,
  `convector` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `ActualData`

CREATE TABLE `ActualData` (
  `id` int(11) NOT NULL,
  `weather_id` int(11) NOT NULL,
  `device_id` int(11) NOT NULL,
  `generation` float NOT NULL,
  `state` tinyint(4) NOT NULL,
  `convector` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `ActualWeather`

CREATE TABLE `ActualWeather` (
  `id` int(11) NOT NULL,
  `city_id` int(11) NOT NULL,
  `date_time_id` int(11) NOT NULL,
  `wind_speed` float NOT NULL,
  `wind_deg` float NOT NULL,
  `clouds` int(11) NOT NULL,
  `temperature` float NOT NULL,
  `sunrise` int(11) NOT NULL,
  `sunset` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `City`

CREATE TABLE `City` (
  `id` int(11) NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `latitude` float NOT NULL,
  `longitude` float NOT NULL,
  `time_zone` int(11) NOT NULL,
  `country_id` int(11) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `Country`

CREATE TABLE `Country` (
  `id` int(11) NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Unm` int(11) NOT NULL,
  `Unf` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `CurrentCapacity`

CREATE TABLE `CurrentCapacity` (
  `id` int(11) NOT NULL,
  `accumulator_id` int(11) NOT NULL,
  `date_time_id` int(11) NOT NULL,
  `capacity` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `Customer`

CREATE TABLE `Customer` (
  `id` int(11) NOT NULL,
  `name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `surname` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `login` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(150) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `email` varchar(80) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(15) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

-- Структура таблиці `DateTimeInfo`

CREATE TABLE `DateTimeInfo` (
  `id` int(11) NOT NULL,
  `day` int(11) NOT NULL,
  `month` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `time` time NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `Device`

CREATE TABLE `Device` (
  `id` int(11) NOT NULL,
  `type_id` int(11) NOT NULL,
  `name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `power` float NOT NULL,
  `voltage` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DeviceParameters`

CREATE TABLE `DeviceParameters` (
  `id` int(11) NOT NULL,
```

```
  `device_id` int(11) NOT NULL,
  `parameter_id` int(11) NOT NULL,
  `value` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DeviceType`

CREATE TABLE `DeviceType` (
  `id` int(11) NOT NULL,
  `type_title` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimCity`

CREATE TABLE `DimCity` (
  `id` int(11) NOT NULL,
  `name` varchar(80) COLLATE utf8mb4_unicode_ci NOT NULL,
  `latitude` float NOT NULL,
  `longitude` float NOT NULL,
  `time_zone` int(11) NOT NULL,
  `country_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimCountry`

CREATE TABLE `DimCountry` (
  `id` int(11) NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `Unm` int(11) NOT NULL,
  `Unf` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimCustomer`

CREATE TABLE `DimCustomer` (
  `id` int(11) NOT NULL,
  `name` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `surname` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `login` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(150) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(80) COLLATE utf8mb4_unicode_ci NOT NULL,
  `phone` varchar(15) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimData`

CREATE TABLE `DimData` (
  `id` int(11) NOT NULL,
  `generation` float NOT NULL,
  `state` tinyint(4) NOT NULL,
  `convector` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimDateTime`

CREATE TABLE `DimDateTime` (
  `id` int(11) NOT NULL,
  `day` int(11) NOT NULL,
```

```
  `month` int(11) NOT NULL,
  `year` int(11) NOT NULL,
  `time` time NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimDevices`

CREATE TABLE `DimDevices` (
  `id` int(11) NOT NULL,
  `name` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL,
  `power` float NOT NULL,
  `voltage` float NOT NULL,
  `type_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimDeviceType`

CREATE TABLE `DimDeviceType` (
  `id` int(11) NOT NULL,
  `type_title` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimForecastData`

CREATE TABLE `DimForecastData` (
  `id` int(11) NOT NULL,
  `forecast_consumption` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimLocation`

CREATE TABLE `DimLocation` (
  `id` int(11) NOT NULL,
  `latitude` float NOT NULL,
  `longitude` float NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `location_type_id` int(11) NOT NULL,
  `city_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimLocationData`

CREATE TABLE `DimLocationData` (
  `id` int(11) NOT NULL,
  `accumulator_capacity` float NOT NULL,
  `consumption` float NOT NULL,
  `fact_sales` float NOT NULL,
  `purchase` float NOT NULL,
  `UA` float NOT NULL,
  `UB` float NOT NULL,
  `UC` float NOT NULL,
  `UAB` float NOT NULL,
  `UAC` float NOT NULL,
  `UBC` float NOT NULL,
  `actual_power` float NOT NULL,
  `balance_load` tinyint(4) NOT NULL,
  `non_critical_load` tinyint(4) NOT NULL,
  `electricity_meter` tinyint(4) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimLocationType`

CREATE TABLE `DimLocationType` (
  `id` int(11) NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `DimWeather`

CREATE TABLE `DimWeather` (
  `id` int(11) NOT NULL,
  `add_date_time` datetime NOT NULL,
  `wind_speed` float NOT NULL,
  `wind_deg` int(11) NOT NULL,
  `clouds` int(11) NOT NULL,
  `temperature` float NOT NULL,
  `sunrise` int(11) NOT NULL,
  `sunset` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `FactData`

CREATE TABLE `FactData` (
  `id` int(11) NOT NULL,
  `date_time_id` int(11) NOT NULL,
  `location_data_id` int(11) DEFAULT NULL,
  `weather_id` int(11) DEFAULT NULL,
  `customer_id` int(11) NOT NULL,
  `location_id` int(11) NOT NULL,
  `data_id` int(11) DEFAULT NULL,
  `device_id` int(11) DEFAULT NULL,
  `forecast_consumption_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `ForecastConsumption`

CREATE TABLE `ForecastConsumption` (
  `id` int(11) NOT NULL,
  `consumption` float NOT NULL,
  `weather_id` int(11) NOT NULL,
  `location_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `ForecastData`

CREATE TABLE `ForecastData` (
  `id` int(11) NOT NULL,
  `weather_id` int(11) NOT NULL,
  `device_id` int(11) NOT NULL,
  `generation` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `ForecastWeather`

CREATE TABLE `ForecastWeather` (
  `id` int(11) NOT NULL,
```

```
  `city_id` int(11) NOT NULL,
  `date_time_id` int(11) NOT NULL,
  `add_date_time` datetime NOT NULL,
  `wind_speed` float NOT NULL,
  `wind_deg` float NOT NULL,
  `clouds` int(11) NOT NULL,
  `temperature` float NOT NULL,
  `sunrise` int(11) NOT NULL,
  `sunset` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `Location`

CREATE TABLE `Location` (
  `id` int(11) NOT NULL,
  `latitude` float NOT NULL,
  `longitude` float NOT NULL,
  `name` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL,
  `city_id` int(11) NOT NULL,
  `customer_id` int(11) NOT NULL,
  `location_type_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `LocationData`

CREATE TABLE `LocationData` (
  `id` int(11) NOT NULL,
  `consumption` float NOT NULL,
  `fact_sales` float NOT NULL,
  `purchase` float NOT NULL,
  `date_time_id` int(11) NOT NULL,
  `location_id` int(11) NOT NULL,
  `UA` float NOT NULL,
  `UB` float NOT NULL,
  `UC` float NOT NULL,
  `UAB` float NOT NULL,
  `UAC` float NOT NULL,
  `UBC` float NOT NULL,
  `actual_power` float NOT NULL,
  `balance_load` tinyint(4) NOT NULL,
  `non_critical_load` tinyint(4) NOT NULL,
  `electricity_meter` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `LocationDevices`

CREATE TABLE `LocationDevices` (
  `id` int(11) NOT NULL,
  `location_id` int(11) NOT NULL,
  `device_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `LocationType`

CREATE TABLE `LocationType` (
  `id` int(11) NOT NULL,
  `type_title` varchar(45) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```
-- Структура таблиці `Logs`

CREATE TABLE `Logs` (
  `id` int(11) NOT NULL,
  `query` varchar(80) COLLATE utf8mb4_unicode_ci NOT NULL,
  `status` int(11) NOT NULL,
  `date_time` datetime NOT NULL,
  `customer_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `Parameters`

CREATE TABLE `Parameters` (
  `id` int(11) NOT NULL,
  `parameter` varchar(30) COLLATE utf8mb4_unicode_ci NOT NULL,
  `device_type_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- Структура таблиці `UserRoles`

CREATE TABLE `UserRoles` (
  `id` int(11) NOT NULL,
  `role_title` varchar(45) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Структура таблиці `Users`

CREATE TABLE `Users` (
  `id` int(11) NOT NULL,
  `name` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `surname` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `login` varchar(30) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(150) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT
NULL,
  `email` varchar(80) NOT NULL,
  `phone` varchar(15) NOT NULL,
  `role_id` int(11) NOT NULL,
  `customer_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=cp1251;

-- Індекси таблиць

-- Індекси таблиці `Accumulator`

ALTER TABLE `Accumulator`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `location_id` (`location_id`),
  ADD KEY `device_id` (`device_id`);

-- Індекси таблиці `ActualData`

ALTER TABLE `ActualData`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
```

```
  ADD KEY `weather_id` (`weather_id`),
  ADD KEY `device_id` (`device_id`);

-- Індекси таблиці `ActualWeather`

ALTER TABLE `ActualWeather`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `city_id` (`city_id`),
  ADD KEY `date_time_id` (`date_time_id`);

-- Індекси таблиці `Admin`

ALTER TABLE `Admin`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `City`

ALTER TABLE `City`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `City_ibfk_1` (`country_id`);

-- Індекси таблиці `Country`

ALTER TABLE `Country`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `CurrentCapacity`

ALTER TABLE `CurrentCapacity`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `accumulator_id` (`accumulator_id`),
  ADD KEY `date_time_id` (`date_time_id`);

-- Індекси таблиці `Customer`

ALTER TABLE `Customer`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `DateTimeInfo`

ALTER TABLE `DateTimeInfo`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `Device`

ALTER TABLE `Device`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `type_id` (`type_id`);

-- Індекси таблиці `DeviceParameters`
```

```
ALTER TABLE `DeviceParameters`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `device_id` (`device_id`),
  ADD KEY `parameter_id` (`parameter_id`);

-- Індекси таблиці `DeviceType`

ALTER TABLE `DeviceType`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `DimCity`

ALTER TABLE `DimCity`
  ADD PRIMARY KEY (`id`),
  ADD KEY `country_id` (`country_id`);

-- Індекси таблиці `DimCountry`

ALTER TABLE `DimCountry`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimCustomer`

ALTER TABLE `DimCustomer`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimData`

ALTER TABLE `DimData`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimDateTime`

ALTER TABLE `DimDateTime`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimDevices`

ALTER TABLE `DimDevices`
  ADD PRIMARY KEY (`id`),
  ADD KEY `type_id` (`type_id`);

-- Індекси таблиці `DimDeviceType`

ALTER TABLE `DimDeviceType`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimForecastData`

ALTER TABLE `DimForecastData`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimLocation`

ALTER TABLE `DimLocation`
```

```
  ADD PRIMARY KEY (`id`),
  ADD KEY `location_type` (`location_type_id`),
  ADD KEY `city_id` (`city_id`);

-- Індекси таблиці `DimLocationData`

ALTER TABLE `DimLocationData`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimLocationType`

ALTER TABLE `DimLocationType`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `DimWeather`

ALTER TABLE `DimWeather`
  ADD PRIMARY KEY (`id`);

-- Індекси таблиці `FactData`

ALTER TABLE `FactData`
  ADD PRIMARY KEY (`id`),
  ADD KEY `dim_location_data_id` (`location_data_id`),
  ADD KEY `dim_date_time_id` (`date_time_id`),
  ADD KEY `dim_weather_id` (`weather_id`),
  ADD KEY `dim_customer_id` (`customer_id`),
  ADD KEY `dim_location_id` (`location_id`),
  ADD KEY `dim_data_id` (`data_id`),
  ADD KEY `dim_device_id` (`device_id`),
  ADD KEY `dim_forecast_consumption_id` (`forecast_consumption_id`);

-- Індекси таблиці `ForecastConsumption`

ALTER TABLE `ForecastConsumption`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `weather_id` (`weather_id`),
  ADD KEY `location_id` (`location_id`);

-- Індекси таблиці `ForecastData`

ALTER TABLE `ForecastData`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `weather_id` (`weather_id`),
  ADD KEY `ForecastData_ibfk_3` (`device_id`);

-- Індекси таблиці `ForecastWeather`

ALTER TABLE `ForecastWeather`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `city_id` (`city_id`),
  ADD KEY `date_time_id` (`date_time_id`);

-- Індекси таблиці `Location`
```

```
ALTER TABLE `Location`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `city_id` (`city_id`),
  ADD KEY `customer_id` (`customer_id`),
  ADD KEY `location_type_id` (`location_type_id`);

-- Індекси таблиці `LocationData`

ALTER TABLE `LocationData`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `date_time_id` (`date_time_id`),
  ADD KEY `location_id` (`location_id`);

-- Індекси таблиці `LocationDevices`

ALTER TABLE `LocationDevices`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `location_id` (`location_id`),
  ADD KEY `device_id` (`device_id`);

-- Індекси таблиці `LocationType`

ALTER TABLE `LocationType`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `Logs`

ALTER TABLE `Logs`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `customer_id_log` (`customer_id`);

-- Індекси таблиці `Parameters`

ALTER TABLE `Parameters`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `device_type_id` (`device_type_id`);

-- Індекси таблиці `UserRoles`

ALTER TABLE `UserRoles`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`);

-- Індекси таблиці `Users`

ALTER TABLE `Users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `id` (`id`),
  ADD KEY `customer_id` (`customer_id`),
  ADD KEY `role_id` (`role_id`);

-- AUTO_INCREMENT для таблиць
```

```
-- AUTO_INCREMENT для таблиці `Accumulator`

ALTER TABLE `Accumulator`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

-- AUTO_INCREMENT для таблиці `ActualData`

ALTER TABLE `ActualData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=505;

-- AUTO_INCREMENT для таблиці `ActualWeather`

ALTER TABLE `ActualWeather`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=337;

-- AUTO_INCREMENT для таблиці `Admin`

ALTER TABLE `Admin`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `City`

ALTER TABLE `City`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

-- AUTO_INCREMENT для таблиці `Country`

ALTER TABLE `Country`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

-- AUTO_INCREMENT для таблиці `CurrentCapacity`

ALTER TABLE `CurrentCapacity`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=337;

-- AUTO_INCREMENT для таблиці `Customer`

ALTER TABLE `Customer`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

-- AUTO_INCREMENT для таблиці `DateTimeInfo`

ALTER TABLE `DateTimeInfo`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=169;

-- AUTO_INCREMENT для таблиці `Device`

ALTER TABLE `Device`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

-- AUTO_INCREMENT для таблиці `DeviceParameters`

ALTER TABLE `DeviceParameters`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=22;

-- AUTO_INCREMENT для таблиці `DeviceType`
```

```
ALTER TABLE `DeviceType`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT для таблиці `DimCity`

ALTER TABLE `DimCity`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimCustomer`

ALTER TABLE `DimCustomer`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimDateTime`

ALTER TABLE `DimDateTime`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimDevices`

ALTER TABLE `DimDevices`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimDeviceType`

ALTER TABLE `DimDeviceType`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimForecastData`

ALTER TABLE `DimForecastData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimLocation`

ALTER TABLE `DimLocation`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimLocationData`

ALTER TABLE `DimLocationData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `DimWeather`

ALTER TABLE `DimWeather`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `FactData`

ALTER TABLE `FactData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `ForecastConsumption`

ALTER TABLE `ForecastConsumption`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=113;
```

```
-- AUTO_INCREMENT для таблиці `ForecastData`

ALTER TABLE `ForecastData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=169;

-- AUTO_INCREMENT для таблиці `ForecastWeather`

ALTER TABLE `ForecastWeather`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=113;

-- AUTO_INCREMENT для таблиці `Location`

ALTER TABLE `Location`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

-- AUTO_INCREMENT для таблиці `LocationData`

ALTER TABLE `LocationData`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=337;

-- AUTO_INCREMENT для таблиці `LocationDevices`

ALTER TABLE `LocationDevices`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

-- AUTO_INCREMENT для таблиці `LocationType`

ALTER TABLE `LocationType`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

-- AUTO_INCREMENT для таблиці `Logs`

ALTER TABLE `Logs`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

-- AUTO_INCREMENT для таблиці `Parameters`

ALTER TABLE `Parameters`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=17;

-- AUTO_INCREMENT для таблиці `UserRoles`

ALTER TABLE `UserRoles`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

-- AUTO_INCREMENT для таблиці `Users`

ALTER TABLE `Users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

-- Обмеження зовнішнього ключа таблиць

-- Обмеження зовнішнього ключа таблиці `Accumulator`

ALTER TABLE `Accumulator`
  ADD CONSTRAINT `Accumulator_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES
`Location` (`id`),
```

```
       ADD CONSTRAINT `Accumulator_ibfk_2` FOREIGN KEY (`device_id`) REFERENCES
`Device` (`id`);

      -- Обмеження зовнішнього ключа таблиці `ActualData`

      ALTER TABLE `ActualData`
       ADD CONSTRAINT `ActualData_ibfk_1` FOREIGN KEY (`weather_id`) REFERENCES
`ActualWeather` (`id`),
       ADD CONSTRAINT `ActualData_ibfk_2` FOREIGN KEY (`device_id`) REFERENCES
`LocationDevices` (`id`);

      -- Обмеження зовнішнього ключа таблиці `ActualWeather`

      ALTER TABLE `ActualWeather`
       ADD CONSTRAINT `ActualWeather_ibfk_1` FOREIGN KEY (`city_id`) REFERENCES `City`
(`id`),
       ADD CONSTRAINT `ActualWeather_ibfk_2` FOREIGN KEY (`date_time_id`) REFERENCES
`DateTimeInfo` (`id`);

      -- Обмеження зовнішнього ключа таблиці `City`

      ALTER TABLE `City`
       ADD CONSTRAINT `City_ibfk_1` FOREIGN KEY (`country_id`) REFERENCES `Country`
(`id`);

      -- Обмеження зовнішнього ключа таблиці `CurrentCapacity`

      ALTER TABLE `CurrentCapacity`
       ADD CONSTRAINT `CurrentCapacity_ibfk_1` FOREIGN KEY (`accumulator_id`)
REFERENCES `Accumulator` (`id`),
       ADD CONSTRAINT `CurrentCapacity_ibfk_2` FOREIGN KEY (`date_time_id`) REFERENCES
`DateTimeInfo` (`id`);

      -- Обмеження зовнішнього ключа таблиці `Device`

      ALTER TABLE `Device`
       ADD CONSTRAINT `Device_ibfk_1` FOREIGN KEY (`type_id`) REFERENCES `DeviceType`
(`id`);

      -- Обмеження зовнішнього ключа таблиці `DeviceParameters`

      ALTER TABLE `DeviceParameters`
       ADD CONSTRAINT `DeviceParameters_ibfk_1` FOREIGN KEY (`device_id`) REFERENCES
`Device` (`id`),
       ADD CONSTRAINT `DeviceParameters_ibfk_2` FOREIGN KEY (`parameter_id`)
REFERENCES `Parameters` (`id`);

      -- Обмеження зовнішнього ключа таблиці `DimCity`

      ALTER TABLE `DimCity`
       ADD CONSTRAINT `country_id` FOREIGN KEY (`country_id`) REFERENCES `DimCountry`
(`id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

      -- Обмеження зовнішнього ключа таблиці `DimDevices`

      ALTER TABLE `DimDevices`
       ADD CONSTRAINT `type_id` FOREIGN KEY (`type_id`) REFERENCES `DimDeviceType`
(`id`);
```

```
      -- Обмеження зовнішнього ключа таблиці `DimLocation`

      ALTER TABLE `DimLocation`
        ADD CONSTRAINT `city_id` FOREIGN KEY (`city_id`) REFERENCES `DimCity` (`id`),
        ADD CONSTRAINT `location_type` FOREIGN KEY (`location_type_id`) REFERENCES
`DimLocationType` (`id`);

      -- Обмеження зовнішнього ключа таблиці `FactData`

      ALTER TABLE `FactData`
        ADD CONSTRAINT `dim_customer_id` FOREIGN KEY (`customer_id`) REFERENCES
`DimCustomer` (`id`),
        ADD CONSTRAINT `dim_data_id` FOREIGN KEY (`data_id`) REFERENCES `DimData`
(`id`),
        ADD CONSTRAINT `dim_date_time_id` FOREIGN KEY (`date_time_id`) REFERENCES
`DimDateTime` (`id`),
        ADD CONSTRAINT `dim_device_id` FOREIGN KEY (`device_id`) REFERENCES
`DimDevices` (`id`),
        ADD CONSTRAINT `dim_forecast_consumption_id` FOREIGN KEY
(`forecast_consumption_id`) REFERENCES `DimForecastData` (`id`),
        ADD CONSTRAINT `dim_location_data_id` FOREIGN KEY (`location_data_id`)
REFERENCES `DimLocationData` (`id`),
        ADD CONSTRAINT `dim_location_id` FOREIGN KEY (`location_id`) REFERENCES
`DimLocation` (`id`),
        ADD CONSTRAINT `dim_weather_id` FOREIGN KEY (`weather_id`) REFERENCES
`DimWeather` (`id`);

      -- Обмеження зовнішнього ключа таблиці `ForecastConsumption`

      ALTER TABLE `ForecastConsumption`
        ADD CONSTRAINT `ForecastConsumption_ibfk_1` FOREIGN KEY (`weather_id`)
REFERENCES `ForecastWeather` (`id`),
        ADD CONSTRAINT `ForecastConsumption_ibfk_2` FOREIGN KEY (`location_id`)
REFERENCES `Location` (`id`);

      -- Обмеження зовнішнього ключа таблиці `ForecastData`

      ALTER TABLE `ForecastData`
        ADD CONSTRAINT `ForecastData_ibfk_1` FOREIGN KEY (`weather_id`) REFERENCES
`ForecastWeather` (`id`),
        ADD CONSTRAINT `ForecastData_ibfk_3` FOREIGN KEY (`device_id`) REFERENCES
`LocationDevices` (`id`);

      -- Обмеження зовнішнього ключа таблиці `ForecastWeather`

      ALTER TABLE `ForecastWeather`
        ADD CONSTRAINT `ForecastWeather_ibfk_1` FOREIGN KEY (`city_id`) REFERENCES
`City` (`id`),
        ADD CONSTRAINT `ForecastWeather_ibfk_2` FOREIGN KEY (`date_time_id`) REFERENCES
`DateTimeInfo` (`id`);

      -- Обмеження зовнішнього ключа таблиці `Location`

      ALTER TABLE `Location`
        ADD CONSTRAINT `Location_ibfk_1` FOREIGN KEY (`city_id`) REFERENCES `City`
(`id`),
```

```
       ADD CONSTRAINT `Location_ibfk_2` FOREIGN KEY (`customer_id`) REFERENCES
`Customer` (`id`),
       ADD CONSTRAINT `Location_ibfk_3` FOREIGN KEY (`location_type_id`) REFERENCES
`LocationType` (`id`);

     -- Обмеження зовнішнього ключа таблиці `LocationData`

     ALTER TABLE `LocationData`
       ADD CONSTRAINT `LocationData_ibfk_1` FOREIGN KEY (`date_time_id`) REFERENCES
`DateTimeInfo` (`id`),
       ADD CONSTRAINT `LocationData_ibfk_2` FOREIGN KEY (`location_id`) REFERENCES
`Location` (`id`);

     -- Обмеження зовнішнього ключа таблиці `LocationDevices`

     ALTER TABLE `LocationDevices`
       ADD CONSTRAINT `LocationDevices_ibfk_1` FOREIGN KEY (`location_id`) REFERENCES
`Location` (`id`),
       ADD CONSTRAINT `LocationDevices_ibfk_2` FOREIGN KEY (`device_id`) REFERENCES
`Device` (`id`);

     -- Обмеження зовнішнього ключа таблиці `Logs`

     ALTER TABLE `Logs`
       ADD CONSTRAINT `customer_id_log` FOREIGN KEY (`customer_id`) REFERENCES
`Customer` (`id`);

     -- Обмеження зовнішнього ключа таблиці `Parameters`

     ALTER TABLE `Parameters`
       ADD CONSTRAINT `Parameters_ibfk_1` FOREIGN KEY (`device_type_id`) REFERENCES
`DeviceType` (`id`);

     -- Обмеження зовнішнього ключа таблиці `Users`

     ALTER TABLE `Users`
       ADD CONSTRAINT `Users_ibfk_1` FOREIGN KEY (`customer_id`) REFERENCES `Customer`
(`id`),
       ADD CONSTRAINT `Users_ibfk_2` FOREIGN KEY (`role_id`) REFERENCES `UserRoles`
(`id`);
     COMMIT;
```

ДОДАТОК В

Лістинги процедур, функцій та запитів

Тригери для додавання записів до сховища даних.

```
CREATE TRIGGER add_dim_customer
after insert on Customer
FOR EACH ROW
INSERT INTO DimCustomer SET
name = NEW.name,
surname = NEW.surname,
login  =  NEW.login,
password = NEW.password,
 email = NEW.email,
 phone = NEW.phone;

CREATE TRIGGER add_dim_date_time
after insert on DateTimeInfo
FOR EACH ROW
INSERT INTO DimDateTime SET
day  = NEW.day,
month = NEW.month,
year  =  NEW.year,
time = NEW.time;

CREATE TRIGGER add_device_type
after insert on DeviceType
FOR EACH ROW
INSERT INTO DimDeviceType SET
type_title  = NEW.type_title;

CREATE TRIGGER add_device
after insert on Device
FOR EACH ROW
INSERT INTO DimDevices SET
type_id  = NEW.type_id,
name = NEW.name,
power = NEW.power,
voltage = NEW.voltage;

CREATE TRIGGER add_country
after insert on Country
FOR EACH ROW
INSERT INTO DimCountry SET
name  = NEW.name,
Unm  = NEW.Unm, Unf  = NEW.Unf;

 CREATE TRIGGER add_device
after insert on Device
FOR EACH ROW
INSERT INTO DimDevices SET
```

```
name = NEW.name,
latitude = NEW.latitude,
longitude = NEW.longitude,
time_zone  = NEW.time_zone,
country_id  = NEW.country_id;

CREATE TRIGGER add_location_type
after insert on LocationType
FOR EACH ROW
INSERT INTO DimLocationType SET
name  = NEW.name;

 CREATE TRIGGER add_location
after insert on Location
FOR EACH ROW
INSERT INTO DimLocation SET
name = NEW.name,
latitude = NEW.latitude,
longitude = NEW.longitude,
location_type_id  = NEW.location_type_id,
city_id  = NEW.city_id;
```

Функція FunctionAddActualLocationData для запису до сховища актуальних даних з датчиків локації.

```
DELIMITER //
CREATE FUNCTION FunctionAddActualLocationData(start_ DATETIME, end_ DATETIME)
RETURNS INT
BEGIN
CREATE TEMPORARY TABLE temp_table SELECT CurrentCapacity.capacity as capacity, LocationData.consumption as consumption, LocationData.fact_sales as sales, LocationData.purchase as purchase, LocationData.UA as UA, LocationData.UB as UB, LocationData.UC as UC, LocationData.UAB as UAB, LocationData.UAC as UAC, LocationData.UBC as UBC, LocationData.actual_power as power, LocationData.balance_load as balance, LocationData.non_critical_load as non_critical, LocationData.electricity_meter as meter, Device.id as device, LocationData.date_time_id as date_time, LocationData.location_id as location, Location.customer_id as customer from CurrentCapacity
JOIN Accumulator ON CurrentCapacity.accumulator_id = Accumulator.id JOIN LocationData ON LocationData.date_time_id = CurrentCapacity.date_time_id
AND LocationData.location_id = (SELECT location_id from Accumulator where CurrentCapacity.accumulator_id=Accumulator.id)
JOIN Device ON Accumulator.device_id = Device.id JOIN Location ON LocationData.location_id = Location.id
JOIN DateTimeInfo ON LocationData.date_time_id = DateTimeInfo.id where DATE_FORMAT(CONCAT(DateTimeInfo.year_, '-', DateTimeInfo.month, '-', DateTimeInfo.day, ' ', DateTimeInfo.time_), '%Y-%m-%d %H:%i:00') BETWEEN start_ and end_ order by DateTimeInfo.id, LocationData.location_id;

FOR var IN (select * from temp_table) do
INSERT INTO DimLocationData(accumulator_capacity, consumption, fact_sales, purchase, UA, UB, UC, UAB, UAC, UBC, actual_power, balance_load, non_critical_load, electricity_meter)
VALUES (var.capacity, var.consumption, var.sales, var.purchase, var.UA, var.UB, var.UC, var.UAB, var.UAC, var.UBC, var.power, var.balance, var.non_critical, var.meter);
```

```
      INSERT INTO FactData (date_time_id, location_data_id, customer_id, location_id,
device_id)
      VALUES (var.date_time, LAST_INSERT_ID(), var.customer, var.location, var.device);
      end for;

         RETURN 1;

      END; //

      DELIMITER ;
```

Функція FunctionAddActualDeviceData для запису до сховища актуальних даних про генерацію енергії для кожного приладу на локації.

```
      DELIMITER //

      CREATE FUNCTION FunctionAddActualDeviceData(start_ DATETIME, end_ DATETIME)
      RETURNS INT
      BEGIN
      CREATE TEMPORARY TABLE temp_table SELECT DATE_FORMAT(CONCAT(DateTimeInfo.year_, '-
', DateTimeInfo.month, '-', DateTimeInfo.day, ' ', DateTimeInfo.time_),
      '%Y-%m-%d %H:%i:00') as actual_date,
      ActualData.generation, ActualData.state, ActualData.convector, ActualWeather.id AS
weather_id, ActualWeather.date_time_id, ActualWeather.wind_speed,
      ActualWeather.wind_deg,      ActualWeather.clouds,      ActualWeather.temperature,
ActualWeather.sunrise, ActualWeather.sunset,
       LocationDevices.location_id, LocationDevices.device_id, Location.customer_id FROM
`ActualData`
      JOIN ActualWeather ON ActualData.weather_id = ActualWeather.id JOIN LocationDevices
ON ActualData.device_id = LocationDevices.id
      JOIN Location ON LocationDevices.location_id = Location.id JOIN DateTimeInfo ON
ActualWeather.date_time_id = DateTimeInfo.id
      where   DATE_FORMAT(CONCAT(DateTimeInfo.year_,   '-',   DateTimeInfo.month,   '-',
DateTimeInfo.day, ' ', DateTimeInfo.time_),
      '%Y-%m-%d %H:%i:00') BETWEEN start_ and  end_ order by actual_date, Location.id,
locationdevices.device_id;

      FOR weath IN (SELECT DISTINCT weather_id FROM temp_table) do
      INSERT INTO DimWeather(add_date_time, wind_speed, wind_deg, clouds, temperature,
sunrise, sunset)
      SELECT actual_date, wind_speed, wind_deg, clouds, temperature, sunrise, sunset
FROM temp_table
      WHERE weath.weather_id = temp_table.weather_id LIMIT 1;

      SET @var_weather:= LAST_INSERT_ID();

      FOR var IN (select * from temp_table WHERE weath.weather_id = temp_table.weather_id)
do
            INSERT INTO DimData(generation, state, convector)
            VALUES (var.generation, var.state, var.convector);

            SET @var_data:= LAST_INSERT_ID();
```

```
            INSERT INTO FactData (date_time_id, weather_id, customer_id, location_id,
data_id, device_id)
            VALUES (var.date_time_id, @var_weather, var.customer_id, var.location_id,
@var_data, var.device_id);
      end FOR;
      end for;

         RETURN 1;

      END; //

      DELIMITER ;
```

Функція FunctionAddForecastData для запису до сховища даних про прогнозовану генерацію енергії для кожного приладу на локації та даних про прогнозоване споживання електроенергії на локації.

```
      DELIMITER //

      CREATE FUNCTION FunctionAddForecastData(start_ DATETIME, end_ DATETIME)
      RETURNS INT
      BEGIN

      CREATE TEMPORARY TABLE temp_devices SELECT ForecastWeather.add_date_time,
ForecastData.generation, ForecastWeather.date_time_id,
      ForecastWeather.wind_speed, ForecastWeather.wind_deg, ForecastWeather.clouds,
ForecastWeather.temperature, ForecastWeather.sunrise,
      ForecastWeather.sunset, ForecastWeather.id as weather,
LocationDevices.location_id, LocationDevices.device_id, Location.customer_id
      FROM `ForecastData` JOIN ForecastWeather ON
ForecastData.weather_id=ForecastWeather.id
      JOIN LocationDevices ON ForecastData.device_id = LocationDevices.id JOIN Location
ON LocationDevices.location_id = Location.id
      JOIN DateTimeInfo ON ForecastWeather.date_time_id = DateTimeInfo.id
      WHERE ForecastWeather.add_date_time BETWEEN
DATE_SUB(DATE_FORMAT(CONCAT(DateTimeInfo.year_, '-', DateTimeInfo.month, '-',
       DateTimeInfo.day, ' ', DateTimeInfo.time_), '%Y-%m-%d %H:%i:00'), INTERVAL 7 HOUR)
AND DATE_FORMAT(CONCAT(DateTimeInfo.year_,
      '-', DateTimeInfo.month, '-', DateTimeInfo.day, ' ', DateTimeInfo.time_), '%Y-%m-
%d %H:%i:00') AND
      DATE_FORMAT(CONCAT(DateTimeInfo.year_, '-', DateTimeInfo.month, '-',
DateTimeInfo.day, ' ', DateTimeInfo.time_), '%Y-%m-%d %H:%i:00')
      BETWEEN "2010-11-01 00:00:00" and "2010-11-01 03:00:00" order BY
ForecastWeather.date_time_id, ForecastWeather.id;

      CREATE TEMPORARY TABLE temp_location SELECT DATE_FORMAT(CONCAT(DateTimeInfo.year_,
'-', DateTimeInfo.month, '-', DateTimeInfo.day, ' ',
      DateTimeInfo.time_), '%Y-%m-%d %H:%i:00') as date_, ForecastWeather.id AS weather,
ForecastConsumption.consumption,
      ForecastConsumption.location_id, ForecastWeather.add_date_time as
add_date_weather, ForecastWeather.date_time_id, Location.customer_id
      FROM `ForecastConsumption` JOIN ForecastWeather ON
ForecastConsumption.weather_id=ForecastWeather.id
```

```
      JOIN Location ON ForecastConsumption.location_id = Location.id JOIN DateTimeInfo
ON ForecastWeather.date_time_id = DateTimeInfo.id
      AND    DATE_FORMAT(CONCAT(DateTimeInfo.year_,    '-',    DateTimeInfo.month,    '-',
DateTimeInfo.day, ' ', DateTimeInfo.time_), '%Y-%m-%d %H:%i:00')
      BETWEEN start_ and end_ order BY ForecastWeather.date_time_id, ForecastWeather.id;


      FOR weath IN (SELECT DISTINCT weather FROM temp_devices) do
      INSERT INTO DimWeather(add_date_time, wind_speed, wind_deg, clouds, temperature,
sunrise, sunset)
      SELECT add_date_time, wind_speed, wind_deg, clouds, temperature, sunrise, sunset
FROM temp_devices
      WHERE weath.weather = temp_devices.weather LIMIT 1;

      SET @var_weather:= LAST_INSERT_ID();

      FOR var IN (select * from temp_devices WHERE weath.weather = temp_devices.weather)
do
            INSERT INTO DimData(generation, state, convector)
            VALUES (var.generation, 1, 1);

            SET @var_data:= LAST_INSERT_ID();

            INSERT INTO FactData (date_time_id, weather_id, customer_id, location_id,
data_id, device_id)
            VALUES (var.date_time_id, @var_weather, var.customer_id, var.location_id,
@var_data, var.device_id);
      end FOR;

      FOR var IN (select * from temp_location WHERE weath.weather = temp_location.weather)
do
            INSERT INTO DimForecastData(forecast_consumption)
            VALUES (var.consumption);

            SET @var_data:= LAST_INSERT_ID();

            INSERT INTO FactData (date_time_id, weather_id, customer_id, location_id,
forecast_consumption_id)
            VALUES (var.date_time_id, @var_weather, var.customer_id, var.location_id,
@var_data);

      end FOR;

      end for;

         RETURN 1;

      END; //

      DELIMITER ;
```

Процедура перенесення даних з бази до сховища.

```
      DELIMITER //
```

```
CREATE PROCEDURE load_data_warehouse(IN start_ DATETIME, IN end_ DATETIME)
 BEGIN

  select FunctionAddActualLocationData(start_, end_);
  DROP TEMPORARY TABLE temp_table;

  select FunctionAddActualDeviceData(start_, end_);
  DROP TEMPORARY TABLE temp_table;

  select FunctionAddForecastData();
  DROP TEMPORARY TABLE temp_devices;
  DROP TEMPORARY TABLE temp_location;

 END;
/

DELIMITER ;


CALL load_data_warehouse('2010-11-01 00:00:00', '2010-11-01 00:30:00');
```

Приклади запитів чотирьох типів для роботи зі сховищем даних.

```
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-',
dimdatetime.day, ' ', dimdatetime.time_),'%Y-%m-%d %H:%i:00') as date_,
dimlocationdata.accumulator_capacity, dimlocationdata.consumption,
dimcustomer.name, dimcustomer.surname, dimlocation.name, dimdevices.name
FROM factdata JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimlocationdata ON dimlocationdata.id = factdata.location_data_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND location_data_id IS NOT NULL;

SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-',
dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00') as date_,
dimcustomer.name, dimcustomer.surname, dimlocation.name, dimdevices.name,
dimdata.generation, dimweather.clouds, dimweather.wind_speed FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimdata ON dimdata.id = factdata.data_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND data_id IS NOT NULL AND
dimweather.add_date_time =
DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day,
' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00');

SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-',
dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00') as date_, dimcustomer.name,
dimcustomer.surname, dimlocation.name, dimdevices.name, dimdata.generation,
dimweather.clouds, dimweather.wind_speed FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
```

```
JOIN dimdata ON dimdata.id = factdata.data_id
JOIN dimdevices ON dimdevices.id = factdata.device_id
WHERE customer_id = 1 AND location_id = 1 AND data_id IS NOT NULL AND
dimweather.add_date_time !=
DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-', dimdatetime.day,
' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00');

SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-',
dimdatetime.day, ' ', dimdatetime.time_), '%Y-%m-%d %H:%i:00') as date_,
 dimcustomer.name,dimcustomer.surname, dimlocation.name, dimweather.clouds,
dimweather.wind_speed FROM factdata JOIN dimdatetime ON dimdatetime.id =
factdata.date_time_id JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimcustomer ON dimcustomer.id = factdata.customer_id
JOIN dimlocation ON dimlocation.id = factdata.location_id
JOIN dimforecastdata ON dimforecastdata.id = factdata.forecast_consumption_id
WHERE customer_id = 1 AND location_id = 1 AND forecast_consumption_id IS NOT NULL;
```

Приклад роботи з видами (View)

```
CREATE VIEW CustomerView AS SELECT name, surname, login, PASSWORD, email, phone
FROM Customer WHERE id = 1;

SELECT * FROM customerview;

CREATE VIEW CustomerForecastLocationData AS
SELECT DATE_FORMAT(CONCAT(dimdatetime.year_, '-', dimdatetime.month, '-',
dimdatetime.day, ' ', dimdatetime.time_),
 '%Y-%m-%d %H:%i:00') as date_, dimweather.clouds, dimweather.wind_speed,
dimweather.temperature, dimweather.sunrise,
dimweather.sunset, dimforecastdata.forecast_consumption FROM factdata
JOIN dimdatetime ON dimdatetime.id = factdata.date_time_id
JOIN dimweather ON dimweather.id = factdata.weather_id
JOIN dimforecastdata ON dimforecastdata.id = factdata.forecast_consumption_id
WHERE customer_id = 1 AND location_id = 1 AND forecast_consumption_id IS NOT
NULL;

SELECT * FROM CustomerForecastLocationData;
```

Приклад php функції для архівування

```
<?php
function data_to_csv(array &$array)
{
   if (count($array) == 0) {
     return null;
   }
   ob_start();
   $df = fopen("php://output", 'w');
   fputcsv($df, array_keys(reset($array)));
   foreach ($array as $row) {
      fputcsv($df, $row);
   }
   fclose($df);
   return ob_get_clean();
}
?>
```